VILNIUS UNIVERSITY

Gabrielė
STUPURIENĖ

Concept-Driven Informatics Education: Extension of Computational Thinking Tasks and Educational Platform for Primary School

DOCTORAL DISSERTATION

Technological Sciences,
Informatics Engineering T 007

VILNIUS 2019

This dissertation was written between 2014 and 2018 at Vilnius University.

Academic supervisor:
Prof. Dr. Valentina Dagienė (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

VILNIAUS UNIVERSITETAS

Gabrielė
STUPURIENĖ

Konceptais grįstas informatikos mokymas: informatinio mąstymo užduočių ir edukacinės platformos išplėtimas pradiniam ugdymui

DAKTARO DISERTACIJA

Technologijos mokslai,
Informatikos inžinerija T 007

VILNIUS 2019

Disertacija rengta 2014–2018 metais Vilniaus universitete.

Mokslinis vadovas:
Prof. Dr. Valentina Dagienė (Vilniaus universitetas, Technologijos mokslai, informatikos inžinerija – T 007)

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor Prof. Dr. Valentina Dagienė as this work would not have been possible without her guidance and support. In the past years she granted me all the freedom to explore my research interests while offering invaluable suggestions that inspired me to stay on course. With her insightfulness in research and passion for the Informatics education field she served as an excellent role model that I aspire to be.

I have great pleasure in acknowledging my gratitude to Prof. Dr. Saulius Gudas (Vilnius University) and Prof. Dr. Vitalij Denisov (Klaipėda University) for very valuable remarks and comments. Their ideas and feedback have contributed significantly to the completion of this dissertation.

I would like to thank all members of the international Bebras community that took part in the development of tasks and in this way influenced the outcome of this dissertation. Especially to Christian Datzko (Switzerland) for collaboration and sharing of ideas.

I am also grateful to the staff of Vilnius University Institute of Data Science and Digital Technologies. Especially I would like to thank all the colleagues in the Educational Systems research group. All of you supported me a lot during my work on the thesis. A special thanks goes to Lina Vinikienė and Dr. Anita Juškevičienė for their spiritual support for whatever I pursue.

I would like to thank my parents and my brother who encouraged me to choose this road and constantly supported me during all these years, also my husband Mindaugas and my children Kajus, Benas and Sofija for listening to my complaints and for believing in me, for the great sacrifices they made in the past years to help me get to this point.

Last but not least, I would like to express my thanks to all the people who have been in one way or another involved in the preparation of this dissertation, for all of their support and tolerance during this challenging period of my life.

What does not kill me makes me stronger.
Friedrich Nietzsche, Twilight of the Idols, 1888

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Computational thinking is the thought process involved in formulating a problem and expressing its solution(s) in such a way that a computer, machine or human can effectively carry out (Wing, 2014).

Concept is defined as an abstract idea which generalizes separate objects, and defines attributes and relations between objects in sciences. It is a mental representation that is implicated in many of human higher thought processes, including various forms of reasoning and inference, categorization, planning and decision making, constructing and testing explanations (Encyclopedia..., 2017).

Conceptual Knowledge is knowledge rich in relationships and understanding. Conceptual knowledge is knowledge of classifications, principles, generalizations, theories, models, or structures pertinent to a particular disciplinary area (Walsh, 2011).

Contest Management System is an essential software environment for running and organizing various contests (IOI-like programming competition, Bebras-like contest, etc.). It should support simple tools, which enable tasks development, user management, grading, announcement area, record of solutions, reports, and data storage (Skūpienė, 2010).

cpm.4.CSE model is a process model that allows the systematic development of the competence model for Computer Science education at University level (Zendler et al., 2016).

Informatics (Information science) is the science that is concerned with the gathering, manipulation, classification, storage, and retrieval of recorded knowledge (American…, 2011).

Informatics/Computer Science/Computing education at school refer to more or less the same thing, that is, the entire scientific discipline underlying the current digitalization and information technology. All of the terms differ greatly from information (and communication) technology (IT, ICT), which mostly focus on computer literacy, that is, knowing how to use computers and their applications as tools (Heintz et al., 2016). The term Informatics education will be used in this thesis.

Informatics concepts play the central role for understanding fundamentals of computers, information technology, software, hardware, and information systems (Dagienė, Stupurienė, 2016b). The use of Informatics concepts is necessary for cognitive processes such as categorization, memory, decision making, learning, and inference.

Informatics concept-driven task is a short task developed for teaching/learning of some subset of Informatics concepts; it requires deep-thinking skills of Informatics field (Dagienė, Stupurienė, 2016a).

Educational platform is a widely-used term used to define the integrity of tools and services for writing, storing, disseminating digital communication, manage student's activities, searching information etc. (Encyclopedia..., 2018). Meanwhile when an existing educational platform is mentioned in this thesis it means the Lithuanian Bebras contest management system.

Student is one who is enrolled or attends classes at a school, college, or university (American…, 2016). It this thesis when student is mentioned it means pupil from school.

ABBREVIATIONS

CAS – organization Computing at School (United Kingdom)

CDIE – concept-driven Informatics education

CDIEM – concept-driven Informatics education model

CMS – Contest Management System

cpm.4.CSE – competence process model for Computer Science education

CT – Computational thinking

EEP – extended educational platform

ICDT – Informatics concept-driven task

MER – model of educational reconstruction

1. INTRODUCTION

1.1 Research Context and Motivation

While Informatics is a well-established discipline in higher education all around the world, it is not the case for secondary and primary education.

From early 2000s what was taught at school was not Informatics as a subject with its own methods, concepts, and principles, but information technology oriented teaching with the goal that the use of software tools was sufficient for students to acquire practical skills (Schwill, 1997; Hadjerrouit, 2009).

During the last decade the situation has been changing. UK Education Secretary Michael Gove in 2012 said (Sutherland, 2013):

> Imagine the dramatic change which could be possible in just a few years... Instead of children bored out of their minds being taught how to use Word and Excel by bored teachers, we could have 11-year-olds able to write simple 2D computer animations… By 16, they could have an understanding of formal logic previously covered only in university courses and be writing their own apps for smartphones.

Informatics education (Fig. 1) is a term that relates more to the practice of teaching/learning about Informatics, rather than the use of information technology in support of teaching and learning.

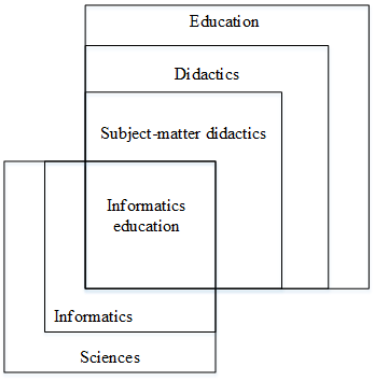


Fig. 1. Context of this study (adapted from Kinnunen (2009))

Informatics is becoming a common, mandatory subject in school curricula of an even increasingly number of countries across the world.

Practically, Informatics is a necessary skill for European students to get the Informatics-intensive jobs of the 21st century. Educationally, Informatics is an invaluable intellectual tool for developing essential conceptual cognitive skills that will serve students through their careers and through all areas of future work (Gander et al., 2013).

Informatics is the only subject that teachers of primary schools have to teach, but never studied themselves (Hromkovic, Lacher, 2017). Teachers need to understand the contributions of Informatics to the understanding of the world and to the growth of intellectual abilities of their students, and that they focus on teaching fundamental and, therefore, stable concepts of Informatics instead of operating instructions for short-term applications.

In order to provide up-to-date Informatics education in schools that integrates every-day experiences of students and thus also activates their intrinsic motivation, current developments and innovations in Informatics must not be neglected. At the same time, general Informatics education needs to focus on central ideas and concepts of Computer Science (Grillenberger et al., 2016).

The concept can be understood as extensive information on a particular object, existing in the human mind. Concepts of Informatics are tightly related with our intentions: what we would like to teach at school. The concept can be defined as a set of objects having common attributes.

Informatics concepts play a central role for understanding fundamentals of computers, information technologies, software, hardware and other devices. However, in practice very often the training of skills in application software is given much more room at schools than to discover and to go deeper into concepts of Informatics.

On the other hand, the digital age demands new abilities, skills and knowledge, which have to be acquired already at school. Overall digitalization of societal processes, public life, economics and health are currently underway. Students should be ready for emerging challenges: big data, virtual reality, artificial intelligence, Industry 4.0, Industry 5.0 and so on.

This thesis is developed as an interdisciplinary research. The work draws upon different research areas. Fig. 2 illustrates the interconnections of the research of this thesis.

First, we see learning theories that are conceptual frameworks describing how students absorb, process, and retain knowledge during learning (Chaudhary, 2013). Cognitive, emotional, and environmental influences as

well as prior experience all play a part in how understanding or a world view is acquired or changed and knowledge and skills retained.

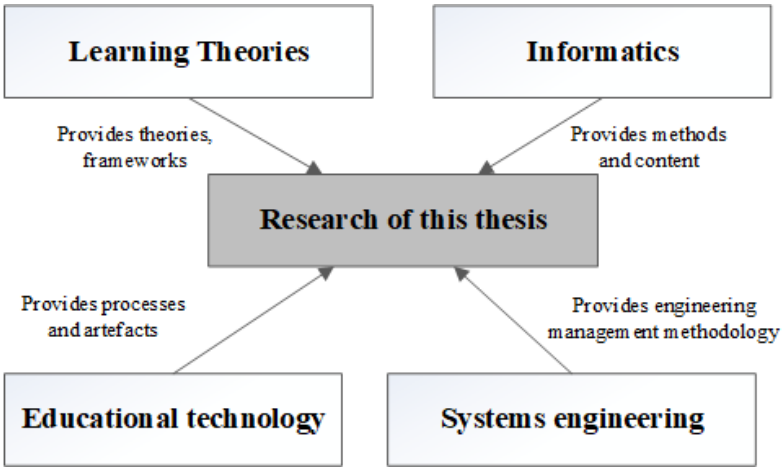


Fig. 2. The areas of research that provide the basis for this thesis

Second there is Informatics, the study of Computer Science including computers design (architecture) and their uses for computations, data processing, and systems control. As a discipline, Informatics spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software (Belford, 2017).

Third we notice the educational technology the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources (Robinson et al., 2008).

The last component is systems engineering that is an interdisciplinary field of general engineering and engineering management with focus on how to design and manage complex systems over their life cycles.

The intersection and interaction of different areas is typical of research of the educational aspects of a specific subject.

In this thesis we present the development of concept-driven Informatics education model for extension of an educational platform. The use of extended educational platform will help learners as well as primary school teachers to acquire competences of Informatics education.

1.2 Objectives and Tasks of the Research

The research object is teaching and learning Informatics at schools, educational processes and competencies of Informatics.

The subject domain is Informatics education at primary school.

The objective of this research is to a develop concept-driven Informatics education model and extend an existing educational platform by adding a well-structured selection of concept learning tasks aligned with the primary school integrated curricula.

In order to achieve this objective, the following research tasks have been stated:

1. To analyze the existing frameworks of basic components and processes for Informatics education and highlight the importance of Informatics concept-driven approach;
2. To develop the concept-driven Informatics education model and adapt it to the primary school integrated curricula;
3. To design a template for learning tasks of Informatics concepts and computational thinking;
4. To develop the educational platform extension model for the concept-driven Informatics education;
5. To construct a prototype of an extended educational platform that implements the model proposed;
6. To evaluate the quality in use of the prototype of an extended educational platform.

1.3 Problem Statement

What should be included in Informatics education in primary school? This problem is raised for the following reasons:

1. There is a widespread controversial idea with a long history that Informatics at schools is only about the use of computers and applications;
2. There is no common agreement (framework) on which part of the background (concepts) of Informatics should be introduced to school, and, in particular, to primary school;
3. There is no educational and technological framework how it should be done and what technologies should be applied.

1.4 Research Methods

A systematic literature review was used to compare, analyze and apply the results of the other researchers. Methodological triangulation in qualitative research that combines content analysis and unstructured interview methods is also applied.

Techniques of data modeling (Entity-Relationship model, UML) and function modeling method (IDEF0) were used to represent the research process. The expert evaluation method and the quality in use model were used to evaluate the proposed model and the quality of extended educational platform.

1.5 Scientific Novelty of the Research

The main novel aspects of concept-driven Informatics education in primary schools suggested in the thesis are as follows:

1. An extended cpm.4.CSE model is adapted for primary school education;
2. The process of identification of Informatics concepts is based on a methodological triangulation in qualitative research;
3. The Informatics concept-driven tasks (ICDT) template (which integrates Informatics concepts with computational thinking skills) was developed and proposed for introducing Informatics at primary school;
4. The educational platform (the Lithuanian Bebras CMS) is extended by a new module containing a specific task selection feature for structured selection of ICDT.

1.6 Statements to be Defended

1. The proposed concept-driven Informatics education model that consists of the extension of cpm.4.CSE model and design of ICDT template is adapted to the primary school integrated curricula. The model is dynamic and can be applied to the other educational levels;
2. The extended educational platform is appropriate and effective (in terms of quality in use) for selection of Informatics concept-driven tasks for Informatics education at primary school.

1.7 Practical Significance of the Results

1. The main practical importance of the work is that the proposed CDIE model can be applied to introduce Informatics for primary school students and herewith to improve primary school teachers' competencies to teach Informatics;
2. The didactic novelty is the paradigm of a short task with a double folded aim: to cover Informatics concepts (together with CT skills) while being solvable in a few minutes to attract students to learn;
3. The educational platform extended by the task selection module allows teachers to structurally and effectively select Informatics concept-driven task for educational process;
4. The prototype of the EEP was highly evaluated by experts according to the chosen quality in use criteria.

1.8 Approval of the Research

The results of the Doctoral thesis were published in 16 scientific publications (8 of them in periodical peer-reviewed journals, and 8 in the proceedings of a scientific conference).

List of Publications:

Articles in the reviewed scientific periodical publications:

1. Dagienė, V., Sentance, S., & **Stupurienė, G.** (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. Informatica, 28(1), 23-44.
2. Dagienė, V., **Stupurienė, G**., & Vinikienė, L. (2017). Implementation of Dynamic Tasks on Informatics and Computational Thinking. Baltic Journal of Modern Computing, 5(3), 306-316.
3. Izu, C., Mirolo, C., Settle, A., Mannila, L., & **Stupuriene, G.** (2017). Exploring Bebras Tasks Content and Performance: A Multinational Study. Informatics in Education, 16(1), 39-59.
4. Benaya, T., Zur, E., Dagiene, V., & **Stupuriene, G.** (2017). Computer Science High School Curriculum in Israel and Lithuania–Comparison and Teachers' Views. Baltic Journal of Modern Computing, 5(2), 164.
5. Dagiene, V., **Stupuriene, G.** (2016). Informatics Concepts and Computational Thinking in K-12 Education: A Lithuanian Perspective. Journal of Information Processing, 24(4), 732-739 (Invited paper).
6. Dagiene, V., & **Stupuriene, G.** (2016). Bebras - A Sustainable Community Building Model for the Concept Based Learning of

Informatics and Computational Thinking. Informatics in Education, 15(1), 25-44.
7. Dagienė, V., Pėlikis, E., **Stupurienė, G.** (2015). Introducing Computational Thinking through a Contest on Informatics: Problem-solving and Gender Issues. Informacijos mokslai, 73, 43-51.
8. Dagienė, V., Pėlikis, E., & **Stupurienė, G.** (2015). Informatinio mąstymo ugdymo užduotys: merginų ir vaikinų sprendimų analizė. Acta paedagogica Vilensia, 35, 53-66.

Proceedings of scientific conferences:

1. Dagienė, V., **Stupurienė, G.** (2018). Short Tasks - Big Ideas: Constructive Approach for Learning and Teaching of Informatics Concepts in Primary Education // Constructionism 2018: Constructionism, computational thinking and educational innovation: conference proceedings. Vilnius: Vilnius University, 169-179. eISBN 9786099576015.
2. Dagienė, V., **Stupurienė, G.,** Vinikienė, L. (2017). Informatics Based Tasks Development in the Bebras Contest Management System. Communications in Computer and Information Science. Vol. 756, 466-477. ISSN 1865-0929, eISSN 1865-0937.
3. Dagiene, V., **Stupuriene, G.,** Vinikiene, L., & Zakauskas, R. (2017). Introduction to Bebras Challenge Management: Overview and Analyses of Developed Systems. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives**.** LNCS, Vol. 10696, 232-243.
4. Dagienė, V., **Stupurienė, G.,** & Vinikienė, L. (2016). Promoting Inclusive Informatics Education Through the Bebras Challenge to All K-12 Students. In Proceedings of the 17th International Conference on Computer Systems and Technologies. ACM. 407-414.
5. **Stupurienė, G.**, Vinikienė, L., & Dagienė, V. (2016). Students' Success in the Bebras Challenge in Lithuania: Focus on a Long-Term Participation. In: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. LNCS, Vol. 9973, 78-89. Springer.
6. Dagienė, V., Futschek, G., **Stupurienė, G.** (2016). Teachers' Constructionist and Deconstructionist Learning by Creating Bebras Tasks // Constructionism in Action 2016, February 1-5, Bangkok, Thailand: conference proceedings. Bangkok: Suksapattana Foundation. 255-263. eISBN 9786169272601.

7. Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., ... & **Stupurienė, G.** (2015). Concepts in K-9 Computer Science Education. In Proceedings of the 2015 ITiCSE on working group reports. ACM, 85-116.
8. Jasutė, E., **Stupurienė, G.** (2015). Finding Threshold Concepts in Computer Science Contest // IFIP TC3 working conference "A new culture of learning: computing and next generations" proceedings. Vilnius: Vilniaus universiteto leidykla, 289-293.

The main results of the thesis were presented and approved at the following scientific conferences.

International conferences and doctoral consortiums:

1. International conference Constructionism 2018: Constructionism, computational thinking and educational innovation. August 21-25, 2018, Vilnius, Lithuania.
2. XIV International symposium on creating and analyzing educational informatics tasks. May 7-11, 2018, Cyprus.
3. 8th International doctoral consortium on informatics engineering education research, December 6-10, 2017, Druskininkai, Lithuania.
4. The 10th International Conference on Informatics in Schools (ISSEP) and doctoral consortium. November 12-15, 2017, Helsinki, Finland.
5. XIII International symposium on creating and analyzing educational informatics tasks. May 29-June 2, 2017, Italy.
6. International conference Constructionism in Action, February 1-5, 2016, Bangkok, Thailand.
7. XII International symposium on creating and analyzing educational informatics tasks. May 21-27, 2016, Turkey.
8. 6th International doctoral consortium on informatics engineering education research. December 08-12, 2015, Druskininkai, Lithuania.
9. 20th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE), July 6-8, 2015, Vilnius, Lithuania.
10. International IFIP TC3 Working Conference "A New Culture of Learning: Computing and Next Generations", July 1-3, 2015, Vilnius, Lithuania.
11. 5th International doctoral school on informatics education and educational software engineering research. November 26-30, 2014, Druskininkai, Lithuania.

1.9 Outline of the Dissertation

The text of the thesis consists of introduction, three main chapters, conclusions, list of references, list of publications and appendixes. The work includes 130 pages of text, 52 figures, 18 tables and 169 references.

Chapter 1 (Introduction) describes the research context, presents the problem statement, discusses motivation, aims and objectives of the research, states research questions, describes research methods, research results, contributions of the thesis. **Chapter 2** presents theoretical backgrounds and related works. **Chapter 3** develops and discusses main results of the research. **Chapter 4** describes experts' evaluation of the quality of the extended educational platform. **Conclusions** present the main results of the dissertation. The structure of the thesis is presented in Fig. 3.

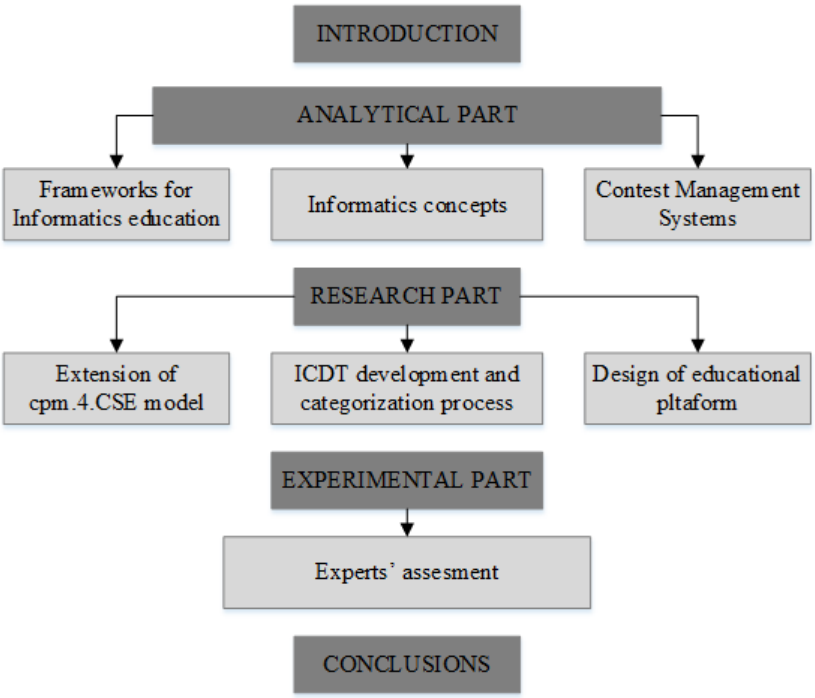


Fig. 3. Structure of the thesis

2. ANALYTICAL PART

The purpose of this chapter is to introduce the theoretical background related to concept-driven approach for Informatics education. Starting with learning theories and approaches, we overview frameworks of basic components and processes for Informatics education. Also, we present common view of related works about Informatics education at primary school.

2.1 Conception of Concept and Learning Approaches

In the physicalism (in philosophy), the concept is a mental representation, which the brain uses to denote a class of things in the world. This is to say that it is literally, a symbol or group of symbols together made from the physical material of the brain (Carey, 2009; Gagne et al., 1993). Concepts are mental representations that allow us to draw appropriate inferences about the type of entities we encounter in our everyday lives (Murphy, 2002). Concepts do not encompass all mental representations, but are merely a subset of them. The use of concepts facilitates the cognitive processes such as categorization, memory, decision making, learning, and inference.

According to Gudavičius (2007), the concept is a global mental unit, and a unit of systematic knowledge about the world. The concept is a result of human being's experience and psychomotor activity (Gudavičius, 2009).

Papaurelytė-Klovienė (2002; 2005) defines the concept as the unit of thought. Concept is all information about something that human being contains in his or her consciousness. According to Gudavičius (2011), the words *meaning*, *notion*, and *concept* describe a certain content of consciousness: image, perception, knowing (Fig. 4).

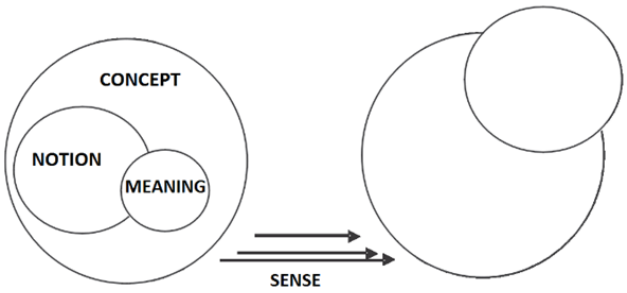


Fig. 4. Meaning – Notion – Concept and Sense (Gudavičius, 2011)

So we can see that concept is understood as a unit of thinking that can be realized in the form of verbalization or as a part of cognition with its own real world image.

Papaurelytė-Klovienė (2004) also emphasizes that when we deal with concept, we cannot avoid the terms of notion and conceptualization. The difference between notion and concept is that notion deals with theoretical knowledge (only basic features are stressed) and cognition (all features are stressed). Conceptualization is also inseparable from concept. It reflects a concept's individuality. Conceptualization is one of the most important processes of cognition performed by human being. The main point of conceptualization is that information in the human mind is processed; concepts, conceptual structures and all conceptual systems lead to in consciousness. In other words, conceptualization is the formation of concepts (Papaurelytė-Klovienė, 2007).

Byrnes and Wasik (1991) noted that conceptual knowledge, which consists of the core concepts for a given domain and their interrelations (i.e., "knowing that"), has been characterized using several different constructs, including semantic nets, hierarchies, and mental models. Procedural knowledge, on the other hand, is "knowing how" or the knowledge of the steps required to attain various goals. Procedures have been characterized using such constructs as skills, strategies, productions, and interiorized actions.

Conceptual understanding appears when children can grasp ideas in a transferrable way and apply them across domains. The ability to transfer skills and knowledge is much more advantageous than memorizing factual information. According to Mills (2016), four salient dimensions of conceptual understanding can be framed: 1) factual and procedural knowledge, 2) making connections, 3) knowledge transfer, and 4) metacognition (Fig. 5).

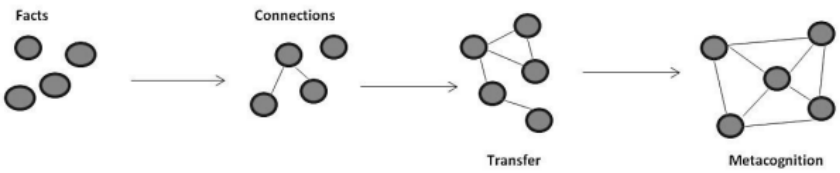


Fig. 5. Dimensions of conceptual understanding (Mills, 2016)

The starting point in the process of conceptual understanding is the attainment of procedural knowledge. Making connections incorporates new

concepts and promotes conceptual learning through concept maps and reflection, and fosters deep learning and enhanced conceptual understanding. The conceptual understanding is deepening by moving forth and back between theory and practice when transferring learning. Despite the fact that both processes of knowledge transfer and the process of making connections seem to be similar, they are two different dimensions of conceptual understanding. It is impossible to transfer knowledge without making connections. Transferring previously learned facts to a new topic helps the learner to reinforce connections and to think in a different way. Finally, metacognition is the knowledge one has about their own thinking and the use of strategies to guide and redirect thinking (Gredler, 2008).

Giddens and Brady (2007) have described conceptual learning as a process by which students learned how to organize information into logical, mental structures. These structures enhance conceptual understanding and strengthen the thought process. Concepts are expressed by words, but learning cannot be accomplished by describing things or processes. The generalization of a concept does not mean creating a description but it means adjusting a physical coordination in the learner's brain (Clancey, 1995).

Deep learning can be encouraged by emphasizing principles and concepts rather than accumulated facts (Hounsell, 1997; Warburton, 2003). Given the unusual breadth of the sustainability agenda, it is important to provide the focus in the form of a unifying framework that permits meaningful dialogue across conventional disciplines. This can be done by identifying key concepts and considering interpretations and implications of each concept in the environmental, social and economic spheres. The key concepts can be reinforced further by returning to them periodically at increasing levels of detail and abstraction – a "spiral curriculum" (Bruner, 1960). It is important to provide a clear structure, a logical progression and unifying themes, and to indicate them at the outset (Entwistle, 1981). Through problem-based learning tasks, students can be encouraged to clarify assumptions, choose analytic techniques and examine value judgments (Hounsell, 1997).

Deep learning is internally motivated and is associated with the intention to understand, rather than to simply pass an assessment task (Marton, Saljo, 1997). Thus, a priority for educators must be to provide an environment where students can develop a strong personal interest in sustainable issues.

Conceptual frameworks should be developed in a clear and graphic fashion. Through enquiry, discussion and problem-based exercises students can make connections between key concepts and visualize these relationships in networks or mind maps.

A concept map or conceptual diagram is a diagram that depicts suggested relationships between concepts (Hager, Scheiber, Corbin, 1997; Mühling, 2014). It is a graphical tool that instructional designers, engineers, technical writers and others use to organize and structure knowledge of a specific domain.

A concept map typically represents ideas and information as boxes or circles, which it connects with labeled arrows in a downward-branching hierarchical structure. The relationship between concepts can be articulated in linking phrases such as causes, requires or contributes to (Novak, Cañas, 2008). Concept maps have become a rather popular tool of teaching, learning and assessment because they are easy to construct and use (Grundspenkis, Strautmane, 2009).

The technique for visualizing these relationships among different concepts is called concept mapping. Concept maps have been used to define the ontology of computer systems, for example with the object-role modeling or Unified Modeling Language formalism (Gonzalez, Dahanayake, 2007).

There are several types of concept maps. One of them is hierarchical structure, where the knowledge domain is allocated on the descending order of importance (Ku, 2007). This type was chosen for use in the thesis.

Concept maps have their origin in constructivism. Constructivism is based on the idea that learners have actively constructed knowledge. Nowadays it is probably the most popular theoretical approach in Informatics education (Machanick, 2007). It derives from the theories of Piaget (Piaget, 1971), who observed learning as occurring in distinct stages, particularly the stages of general understanding which a child went through (Piaget, 1953). The simpler type of learning is often referred to as assimilation when new details fit into the existing model. Learning that requires changes to the model is referred to as accommodation (von Glasersfeld, 1995).

Papert extended the Piagetian theory of constructivism in a way that applies to practical construction and named it by constructionism (Papert, 1987), it specifies how individual learners construct mental models in order to understand the world around them.

For both constructivism and constructionism, knowledge is built by the learner instead of being presented and imposed on students by an expert, such as a teacher. Constructionism adds to the constructivist perspective the idea of artifact construction. Where constructivists view the learner as an active builder of knowledge, constructionism places a critical emphasis on

having learners engage in artifacts constructions that are external and shared. In contrast to Piaget who focuses on cognitive processes of learning, Papert's constructionism focuses on learning through making and emphasizes individual learners' interactions with their artifacts that are mostly built through the assistance of digital media and computer based technologies (Parmaxi, Zaphiris, 2014).

Wilensky (1991) took this point further providing a new perspective into our understanding of concrete elucidation, that concreteness is not a property of an object but rather a property of a person's relationship to an object. Concepts that were hopelessly abstract at one time can become concrete for us if we get into the "right" relationship with them. In light of this perspective, any idea, concept or a piece of knowledge can become concrete provided that a person develops a set of representations, interactions and connections with them.

Constructionism provides us with the basic idea of an appropriate learning object. Such an object should support a learner's step-by-step understanding of the materials and concepts it represents, allowing the learner to self-construct his or her own knowledge. Constructionism is focused on the personal construction of ideas and relations through the construction of real-life artifacts (Ben-Ari, 2001).

2.2 Overview of Frameworks for Informatics Education

A number of frameworks of basic components and processes for Informatics education are described in literature. Some of them came from university level or other sciences areas.

One of them is a model of educational reconstruction (MER) that was developed by Kattmann et al. (1996). They argue that the central aspects of lesson planning such as the perspectives of learners are often only considered after the clarification and analysis of the science subject matter, if considered at all. They saw a clear gap between science education research and science instruction practice. However, as Diethelm et al. (2012) point out that Informatics differs from other science subjects in goals, knowledge structure and teaching methods. Therefore, they extended the original ideas with missing aspects from MER and also take into account the general educative nature of Informatics education in schools. Therefore, they have adapted MER for CS education (MER-CSE) (Fig. 6) and illustrated some of the components with examples.

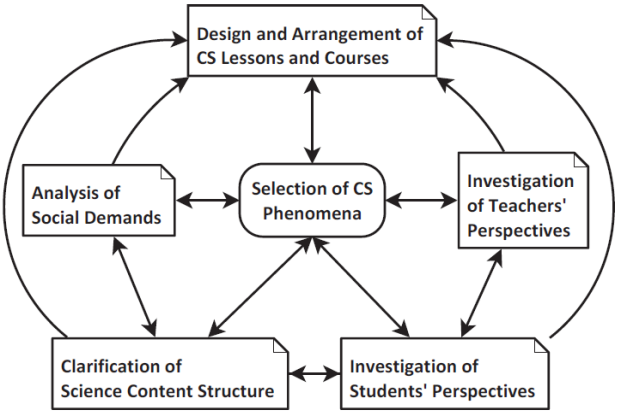


Fig. 6. Educational Reconstruction for Computer Science Education (Diethelm et al., 2012)

The authors highlighted the role of context and phenomena "to motivate the students, to open connections to prior knowledge or to show application situations of the intended knowledge." This approach also ties in with the ideas of Piaget's constructivism, i.e. that learning means to build knowledge structures from interpreting new information based on existing knowledge and experience (Diethelm et al., 2012).

Later, Grillenberger et al., in 2016 proposed the idea that when preparing the contents of innovative Informatics topics for schools, merely reducing the complexity and perceived difficulty of the subject matter is not enough. Instead, the field needs to be thoroughly examined. Innovations in Informatics can be didactically prepared for teaching by using the model of educational reconstruction for CSE (MER-CSE). Therefore, the authors described the adaptation and application of the MER-CSE as a research framework (Fig. 7).

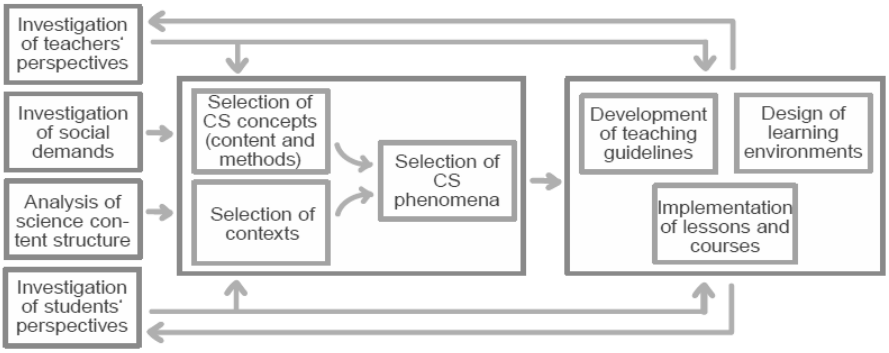


Fig. 7. Application of the MER for CS Education (Grillenberger et al., 2016)

The selection of concepts helps us to focus on aspects that are interesting for students and at the same time represent fundamentals of the subject. Students' perceptions are important here because they tell us about their "mental constructions" with regard to the content in question, which will affect the choice and preparation of concepts for contextualized learning (Grillenberger et al., 2016; Grillenberger, Romeike, 2017).

The next framework is process-based development of competence models to Computer Science (Informatics) education at university level, which is provided by Zendler, Seitz and Klaudt (Zendler et al., 2016). The process model (cpm.4.CSE) includes eight subprocesses: A1 determines competence concept; A2 determines competence areas; A3 identifies Computer Science concepts; A4 assigns competence dimensions to Computer Science; A5 codes competences; A6 formulates competences; A7 formulates learning tasks; and A8 formulates test tasks (Fig. 8).

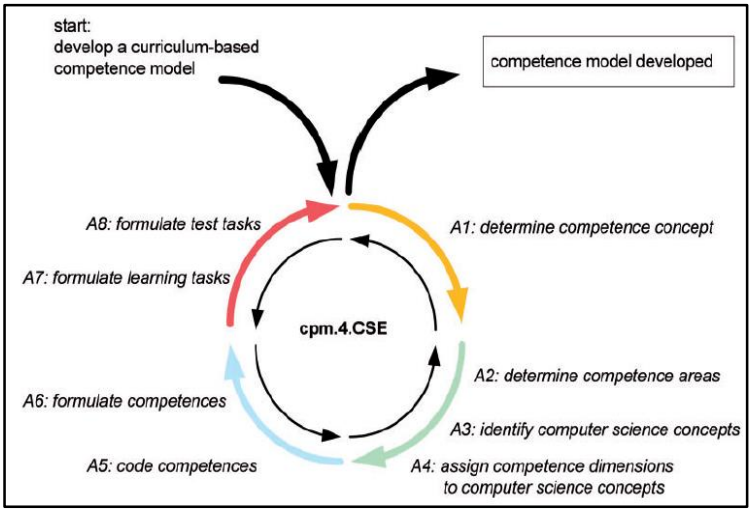


Fig. 8. Competence process model for Computer Science education (Zendler et al., 2016)

The model is based on four main dimensions of competence (Rychen, Salganik, 2003): (1) knowledge, (2) cognitive skills, (3) practical skills, and (4) attitudes.

Before developing and presenting this cpm.4.CSE model, Zendler and colleagues empirically determined four competence areas for high school education on the basis of expert assessments (Zendler, Spannagel, 2008; Zendler et al., 2014). Using a cluster analysis approach and with multidimensional scaling, the following competence areas have been

identified: (1) information technology, (2) modeling, (3) computer communication, and (4) software engineering.

The last framework analyzed was suggested by Manev and Maneva (2017). They proposed methodology for development of school curricula in Computing. The main feature of this methodology is that it is extracted from the guidance for creating university Computer science curricula (CS2013) of the most respected professional associations in the domain – ACM (Association for Computing Machinery) and the Computer Science section of IEEE (The Institute of Electrical and Electronics Engineers).

The body of knowledge of the Computing domain is hierarchically organized in four levels. Level 1 contains the fields of the domain; on Level 2 the fields are divided into areas; on Level 3 each area is divided into units; on Level 4 each unit is composed of individual topics.

According to Manev and Maneva (2017), in the secondary school model students can have only one or maximum two courses in the domain with one to four class hours per week; asking for more class hours for Computing nowadays seems not realistic. So the model has to include some "class hours per week" scheme, which defines the grade, number of courses, class hours per week and distribution of the class hours. The body of knowledge for the created curriculum was chosen mainly from two fields of Computer Science and Information Technology from Computing Curricula (2001). Is it possible to define different kind of models also on the base of preferred main fields – CS-oriented (most appropriate for mathematical and engineering schools), IT-oriented (more appropriate for language, art and sport schools), CS & IT-oriented (more appropriate for regular schools).

All above mentioned frameworks are compared in Table 1.

Table 1. Comparison of the frameworks

Framework \ Criterion	Clearly defined	Emphasizes importance of concepts	Emphasizes importance of competences	Composition of lesson and course
Diethelm et al., 2012	No	No	No	Yes
Grillenberger et al., 2016	Yes	Yes	No	Yes
Zendler et al., 2016	Yes	Yes	Yes	No
Manev, Maneva, 2017	No	No	No	Yes

The most important criteria of comparison are whether the framework is clearly defined (steps, processes, relations), or whether it emphasizes the importance of Informatics concepts, the importance of Informatics competences, and whether it describes the composition of lesson and course.

The overview of these frameworks lets us get an understanding of basic components and processes for Informatics education. It was decided to choose one of them, therefore we selected cpm.4.CSE.

These are the reasons why this framework was selected:

1. In this framework the entire educational process has clearly indicated steps;
2. From our view of point, it is also important to determine Informatics competencies and identify Informatics concepts.
3. This framework is closely related with what Informatics topics should be taught at the university level, so that they can be adapted to the school level;

2.3 Concepts for Informatics Education

In 2013, the Association Informatics Europe and ACM Europe Working Group on Informatics Education prepared the report "Europe cannot afford to miss the boat" (Gander et al., 2013). Based on the analysis of the current situation of Informatics education in Europe and experience in many countries, this report makes four key recommendations:

1. All students should benefit from education in digital literacy, starting from an early age and mastering the basic concepts by age 12. Digital literacy education should emphasize not only skills but also the principles and practices of using them effectively and ethically.
2. All students should benefit from education in Informatics as an independent scientific subject, which is studied both for its intrinsic intellectual and educational value and for its applications to other disciplines.
3. A large-scale teacher training program should urgently be started. To bootstrap the process in the short term, creative solutions should be developed involving school teachers paired with experts from academia and the industry.
4. The definition of Informatics curricula should rely on the considerable body of existing work on the topic and the specific recommendations of the present report.

For better understanding in which direction the research concerning the Informatics concepts education is going, it was decided to make a revision of research publications by using a systematic literature review. A systematic review is a structured, comprehensive, transparent, and methodical process in which literature is rigorously identified, appraised, and synthesized (Kitchenham et al., 2004; Biolchini et al., 2005). This review was conducted in 2016 and later supplemented.

To gain a comprehensive understanding of current literature on a topic and identify literature gaps, it was looked to the literature to answer the question: What evidence in the literature is in order to determine what kind of concepts does exist in Informatics education at school?

Based on the principles of the systematic literature review, first, the aim of this analysis, selected electronic sources and search terms were determined. Suitable literature was collected according abstract and later according the full text of papers.

The selected electronic sources are: Thomson Reuters Web of Science, SpringerLink, ACM Digital Library, Ebsco Host, Google Scholar.

Years covered by search: 2005– 2016, language – English.

The search was conducted according to three Boolean search terms:

1. Concept* AND computer science education AND school*

335 pieces of literature were found, only 20 suited our analysis.

2. Concept* AND informatics education AND school*

68 pieces of literature were found, only 3 suited for analysis.

3. Concept* AND computing education AND school*

67 pieces of literature were found, only 2 suited for analysis.

The topic related with Informatics concepts education at school was not very popular in academic electronic resources. More information can be found in the existing curricula of various countries (Italy, Poland, the Netherlands, the United Kingdom).

In Informatics education research, there is a strong consensus that teaching should focus on aspects that are fundamental to the subject and relevant in the long term instead of short-lived technical developments. For this reason, various catalogs of principles, ideas and concepts, which characterize Informatics or one of its areas, have been proposed over the past 30 years (Grillenberger, Romeike, 2017).

After the systematic literature review of appropriate scientific literature and curricula related with Informatics education at school, we can notice that there exist three types of concepts:

1. Informatics concepts;

2. Computational thinking concepts;
3. Programing concepts.

Programming concepts are part of Informatics concepts, so they not will be described as separate concepts. So the two types of concepts are discussed in detail below.

2.3.1 Informatics Concepts

Computer Science Teacher Association (CSTA) from the USA in 2003 provides a Model Curriculum for K-12 Computer Science (Tucker et al., 2003). For the purposes of that document, they rely heavily on the definition of computer science and believe that this definition has the most direct relevance to high school Computer Science education. They define the discipline as follows:

"Computer science is the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (p. 6).

In 2006 they improved the Model Curriculum for K-12 Computer Science (Tucker et al., 2006) and provided 14 topics. All topics are described in detail in Verno et al. (2006). In 2011 the Model Curriculum for K-12 Computer Science was revised one more time and was called K-12 Computer Science Standards.

Strands in Computer Science standards:

1. Computational thinking;
2. Collaboration;
3. Computing practice and programming;
4. Computers and communication devices;
5. Community, global, and ethical impacts.

In 2012 the organization Computing at School (CAS; UK) prepared the document "Computer Science: A Curriculum for Schools". In this document Computer Science is defined as a discipline that seeks to understand and explore the world around us, both natural and artificial, in computational terms. Computer Science is particularly, but by no means exclusively, concerned with the study, design, and implementation of computer systems, and understanding the principles underlying these designs.

A number of key concepts are grouped:

1. Languages, machines, and computation;
2. Data and representation;
3. Communication and coordination;
4. Abstraction and design;

5. The wider context of computing.

Mark Dorling and Matthew Walker (2014), from Computing at School, proposed the Computing Progression Pathways, which describes how it can be used to acknowledge progression and reward performance in mastering both the computing program of study content and computational thinking skills. It includes the dependencies and interdependencies between concepts and principles. This may help non-specialist teachers and inexperienced teachers to understand what should be taught in the classroom (Selby, Dorling, Woollard, 2014; Selby, 2014):

The framework is grid-based. Each row represents a level of student progression. Six strands are represented as columns:

1. Algorithms;
2. Programming & Development;
3. Data & Data Representation;
4. Hardware & Processing;
5. Communication & Networks;
6. Information Technology.

A group of researchers (the author of this thesis was a member of the group) have conducted the research regarding concepts in K–9 Computer Science Education (Barendsen et al., 2015) and present the results of the exploratory study. They were interested in the CS content in K-9, i.e., topics and ideas belonging to the subject matter, regardless of the specific skills or attitudes in which they appear and referred to these topics and ideas as concepts. They have clustered the knowledge areas into a conveniently small number of categories suitable to classify the CS content for K-9 education, providing enough detail to distinguish variations in content. This report presents the results of this exploratory study.

The documents analyzed in this preliminary report were:

1. CSTA curriculum, K-9 part;
2. CAS curriculum, K-9 part;
3. English (EN) national curriculum, K-9 part;
4. Italian (IT) guidelines, K-8 part.

The distribution of code occurrences found in the documents is displayed in Table 2.

Table 2. Occurrences of codes within the knowledge categories

	CSTA	CAS	EN	IT
Algorithms	14	36	15	16
Engineering	13	13	1	0
Architecture	12	26	5	3
Society	10	2	3	3
Programming	9	17	7	6
Intelligence	6	1	0	0
Modelling	6	0	2	3
Mathematics	5	1	1	8
Security	5	2	1	1
Data	2	24	7	13
Graphics	2	0	0	0
Networking	2	32	7	0
Usability	1	0	1	0
Rest	0	0	0	0
Total	*87*	*154*	*50*	*53*

These absolute numbers reflect the respective sizes of the documents. For example, the English and Italian documents are written in a more compact style than the CAS curriculum. The global concept distribution suggests that all four K-9 documents give substantial attention to algorithmic aspects, especially CAS, EN and IT. Programming is seen in the documents in comparable fractions. The engineering aspect is absent in the Italian guidelines, and does not play an important role in EN either. CSTA seems to have more emphasis on societal aspects than the other two documents. For instance, in CAS, societal aspects are not very prominent, in favor of the more technical aspects (Engineering, Networks). These categories appear to be the main differences between CAS and EN.

In 2016 CSTA, ACM, and Code.org joined forces with more than 100 advisors within the computing community and prepared the K–12 Computer Science framework (K-12 Computer Science..., 2016). The framework identified the key K-12 Computer Science concepts and practices which students expect to know in grades 2, 5, 8, and 12. Beginning with the earliest grades and continuing through the 12th grade, students will develop a foundation of Computer Science knowledge and learn new approaches to problem solving that harness the power of computational thinking to become both users and creators of computing technology. By applying Computer Science as a tool for learning of various disciplines, students will actively participate in the world that is increasingly influenced by technology.

The core concepts of the K–12 Computer Science Framework represent the major content areas in the field of Computer Science. The core concepts are delineated by multiple subconcepts that represent specific ideas within

each concept. The learning progressions for each subconcept provide a thread connecting students' learning from kindergarten to the 12th grade.

Core concepts of the framework are as follows:

1. Computing Systems;
2. Networks and the Internet;
3. Data and Analysis;
4. Algorithms and Programming;
5. Impacts of Computing.

Crosscutting concepts are themes that illustrate connections among different concept statements. They are integrated into concept statements, instead of existing as an independent dimension of the framework. The crosscutting concepts that are represented in each concept statement are noted in the statement's descriptive material.

Crosscutting concepts of the framework:

1. Abstraction;
2. System Relationships;
3. Human–Computer Interaction;
4. Privacy and Security;
5. Communication and Coordination.

The practices of the K–12 Computer Science Framework are the behavior that computationally literate students use to fully engage with the core concepts of Computer Science. Concepts and practices are integrated to provide complete experiences for students engaging in Computer Science. The criteria of the selection of practice should be the following:

1. help students engage with course content through the development of artifacts;
2. be helpful to fully explore and understand the framework concepts;
3. capture important behaviors that computer scientists engaged in;
4. be based on processes and proficiencies with importance in Computer Science.

The practices intentionally overlap with those in other disciplines and use similar language to help teachers make connections between Computer Science and other disciplines they are more familiar with and to make the framework more accessible to a wider audience.

The seven core practices of Computer Science describe the behavior and ways of thinking that computationally literate students use to fully engage in today's data-rich and interconnected world.

The new K-12 computing curriculum draft for Chinese Taipei (Taiwan Province) secondary schools was designed to launch in 2018, but the draft

only outlined themes and contents for students to learn, without further details about the key concepts to be covered in the contents (Hu et al., 2017). Therefore, in 2016, the Delphi study was conducted to survey the opinions about what "key learning concepts" should be included for the implementation at the secondary school level based on the draft. By adopting the Delphi method, different viewpoints of computer scientists and secondary school computing teachers were collected to build consensus of key concepts through a series of convergence. This study found computer scientists tended to be more conservative about this issue, therefore they suggested that the advanced and theoretical concepts are not essential at the secondary level, e.g., recursion, searching, sorting, data compression, data conversion. This was because the computer scientists considered these concepts as they were when they had studied at college. Computing teachers knew how to simplify these concepts for teaching at the secondary level. In the Delphi study the following six categories of learning contents were described: 1) programming; 2) algorithm design; 3) system platform; 4) data representation, processing and analysis; 5) application of ICT; 6) ICT and social, legal and ethical issues.

After the overview of literature related with Informatics concepts it was decided to match proposed categories of core Informatics concepts (Table 3).

Table 3. Matching of core categories of Informatics concepts

Key concepts (CAS, 2012)	**Six strands (CAS, 2014)**	**Core concepts of the framework (K-12 CS Framework, 2016)**	**Six categories of learning contents (Taiwan, 2017)**
Languages, machines, and computation	Algorithms AND Programming and development	Algorithms and Programming	Programming AND Algorithm design
Data and representation	Data and Data Representation	Data and Analysis	Data representation, processing and analysis
Abstraction and design	Hardware and Processing	Computing Systems	System platform;
Communication and coordination	Communication and Networks	Networks and the Internet	
The wider context of computing		Impacts of Computing	ICT and social, legal and ethical issue
	Information Technology		Application of ICT

The results show that when comparing the four documents with the frameworks of Informatics concepts categories, three main categories were suggested:

1. Algorithms;
2. Programming issues;
3. Data and representation.

Other categories are not so clearly distinguished and depend on various interpretations.

2.3.2 Concepts of Computational Thinking

The term computational thinking (CT) was popularized in 2006 with Jeanette Wing's article (2006) but actually originated with Seymour Papert's constructionist learning ideas (1996). There are differences between these two definitions: Wing's definition is more focused on problem solving and Papert's definition is more focused on ideas and analysis (Mannila et al., 2014). Subsequent research has expanded and interpreted the term further (Grover & Pea, 2013; Kalelioglu et al., 2016; Lu & Fletcher; 2009, Selby & Woollard, 2013; Wolz et al., 2011, Lee et al., 2014).

In the summer of 2009, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) began a multi-phase project aimed at developing an operational definition of computational thinking for K-12 (Barr, Stephenson, 2011). They identified many ideas about what computational thinking is and what it could be in classrooms. When challenged with the task of describing what makes computational thinking differ from other kinds of thinking, participants tended to focus on the centrality of the computer and a set of concepts that computational thinking and doing encompass:

> CT is an approach to solving problems in a way that can be implemented with a computer. Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. CT is a problem solving methodology that can be automated and transferred and applied across subjects.

The operational definition provides a framework and vocabulary for computational thinking that will resonate with all K–12 educators (International Society…, 2011).

Computational thinking is a problem-solving process that includes (but is not limited to) the following characteristics:

1. Formulating problems in a way that enables us to use a computer and other tools to help solve them;

2. Logically organizing and analyzing data;
3. Representing data through abstractions such as models and simulations;
4. Automating solutions through algorithmic thinking (a series of ordered steps);
5. Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources;
6. Generalizing and transferring this problem solving process to a wide variety of problems.

According to Computing at School (2012), computational thinking is the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from computing to understand and reason about both natural and artificial systems and processes. Computer Science is more than programming, but programming is one of the absolutely central processes for it. Programs written with different syntax can perform the same semantic task. In an educational context, programming encourages creativity, logical thought, precision and problem-solving, and helps foster the personal, learning and thinking skills required in the modern school curriculum. Programming gives concrete, tangible form to the idea of "abstraction", and repeatedly shows how useful it is.

1. Abstraction:
    - Modeling (the process of developing a representation of a real world issue, system, or situation that captures the aspects of the situation that are important for a particular purpose, while omitting everything else);
    - Decomposition (a problem can often be solved by decomposing it into sub-problems, solving them, and composing the solutions together to solve the original problem);
    - Generalization (the process of recognizing these common patterns, and using them to control complexity by sharing common features).
2. Programming:
    - Designing and writing programs;
    - Abstraction mechanisms (effective use of the abstraction mechanisms supported by programming languages (functions,

procedures, classes, and so on) is central to managing the complexity of large programs);

- Debugging, testing, and reasoning about programs.

Focusing only on mental processes, Selby and Woollard (2013) define CT as a cognitive or mental process of humans, not of machines, of problem solving in a broad sense, and involving abilities such as:

1. Abstraction consists of hiding the inherent complexity of reality to represent only its essential aspects;
2. Decomposition consists of dividing a task or problem into simpler parts so that they can be solved;
3. Algorithmic thinking consists of defining a task as a set of simple step-by-step instructions;
4. Evaluation consists of assessing the advantages and limitations of a solution;
5. Generalization consists of being able to move from a specific situation to more general ones.

Computational thinking is at the heart of the Computer Science practices and is delineated by practices from the K–12 Computer Science Framework (K-12 Computer Science..., 2016):

1. Recognizing and defining computational problems;
2. Developing and using abstractions;
3. Creating computational artifacts;
4. Testing and refining computational artifact.

CT definitions analysis was conducted by Juškevičienė and Dagienė (2018), and presented in the percentage form of the words used to describe the essence of CT: problem solving (22%), abstraction (13%), computer (13%), process (9%), science (7%), data (7%), effective (6%), algorithm (6%), concepts (5%), ability (5%), tools (4%) and analyzing (4%). However, some researchers concluded that current limitations in the CT definition are that it is shaped by technology-aided problem solving (Haseski et al., 2018).

Flórez et al., (2017) mentioned that it is important to understand the complexity and importance of teaching CT, and differentiate among specific key terms: computer programming, computational thinking, and algorithmic thinking. They define computer programming as the process through which a person is able to provide a set of instructions that will communicate, as specifically and accurately as possible, a procedure, method, practice, or task to a machine. They also define algorithmic thinking as a way of obtaining a solution through a series of steps. Thus, CT is a broader term that involves

among other skills, algorithmic thinking, logic, abstraction, generalization, decomposition, and debugging.

The other important aspect is that CT is related not only with Informatics or programming, but also with other disciplines. Sengupta, Dickes and Farris (2018) highlight the importance of grounding computational thinking in representational and epistemic practices that are central to knowing and doing in science, and more broadly, in STEM education.

Computational thinking and digital competence are indicated by many education policy makers as important twenty-first century skills. The European Commission Science Hub has promoted computational thinking and has launched the Digital Competence Framework 2.0 (DigCom)[1] in its portal. Nowadays computational thinking and digital competence are essential skills and the young generation should learn them for life (Juškevičienė, Dagienė, 2018).

2.4 Informatics Education at Primary School across the World

Informatics education is an emerging area starting with the first level in primary schools. Informatics activities can be included in other subjects but not only at the level of using digital technologies.

In particular, there are two major educational challenges related to: (a) what Informatics content to teach across different educational levels, and (b) what body of knowledge do teachers need to have to be able to teach the Informatics curriculum (Angeli et al., 2016).

There are many reasons for including Informatics education at the primary level. One of them is reducing gender inequality in the information technology sphere. Upper school students already have a vision on what is "for girls" and what things are "for boys". Informatics usually falls into "for boys only" category. This problem might be partly avoided by introducing the course earlier (Margolis, Fisher, 2003).

Informatics education researchers also have concerns with regard to teaching Informatics at primary school. These concerns are primarily linked to the incompatibility between abstraction, an essential process in Informatics, and children's weakness to understand abstraction because of their very young age. Armoni (2012) explained that abstraction is an inherent component of Informatics that is always encapsulated during the

[1] https://ec.europa.eu/jrc/en/digcomp/digital-competence-framework

process of thinking about and automating a solution to a problem. From a Piagetian perspective, children before the age of seven cannot really understand concrete logic, whereas children between seven and eleven years old can solve problems that apply to concrete objects, but not problems that apply to abstract concepts or phenomena. Conversely, Gibson (2012) argued that high school is too late for exposing students to Informatics for the first time, and stated that early exposure at kindergarten is necessary. He found that young children can think abstractly when concrete reference systems are used to situate their thinking.

The numerous studies have confirmed the benefits generated by the teaching of programming concepts as they require the use of structured thinking and in the development of basic cognitive skills, which are associated, for example, with the mathematical ability and the development of logical thinking in children of preschool and early primary school age (Kazakoff and Bers, 2012; Grover and Pea, 2013; Kazakoff et al., 2013; Strawhacker et al., 2015).

In 2010, Austria had a project "Informatik erLeben" (Experiencing Informatics) that aims at attracting students to Informatics as a constructive, technical discipline (Mittermeir et al., 2010; Bischof, Sabitzer, 2011). Students from primary school up to upper secondary school obtained lectures by university teachers spread over a period of one and a half year. The prepared lessons were proposed to students and the selected core-concepts of Informatics were introduced in a playful way at an age-specific level. The topics are divided into core-concepts and into several modules that can be composed individually. For example, Coding (Morse Game; Creating a Code with Colors; Code trees; Error Detection); Computer Networks (Chinese Whispers; Communication Rules; Postman-Game); Algorithms (Instructions how to get somewhere); Sorting (Binary Search-tree); Searching (Blind Search; Searching in a linear Structure) etc.

Depending on the topic they act either as part of the computer, serving as data or as an object being manipulated by algorithms, or assuming some role of a program. On principle, computers were specifically not used during the lessons. The students learned, based on activities, simulations, and animations. Important didactical principles behind the concept are discovery learning and teamwork.

Based on the project reflection there are some useful findings:

1. It is very important to start at an early age to broaden the students' image of Informatics and to create interest;

2. While some boys already have been interested in Informatics before, all participating girls could be influenced;
3. Students must have the possibility to attend exciting Informatics lessons during all grades. Because primary school kids are very open and enthusiastic about new topics and concepts, it is necessary to bring more technical topics in all primary schools.

Duncan and Bell (2015), having established the six general areas covered by existing primary school curricula, analyzed three key English-language computing curricula: the CSTA K-12 Computer Science standards (2011), the English computing curriculum (2014), and the Australian Digital Technologies curriculum (2013). They found some notable features:

1. all three curricula introduce programming concepts from the first year (5 or 6 years old), using only sequences and turtle graphics, which are based on concrete physical motion that students can relate to;
2. selection (branching) and IF iteration (repetition) are introduced from about seven years of age, it seems to be in the form of simple WHILE-DO counted loops. More sophisticated iteration with conditions on the loops, and the introduction of textual (general purpose) languages, seems to be expected around 11 or 12 years of age;
3. topics relating to safety and ethics are covered from the very first year, again gradually increasing in sophistication from simple scenarios for young students to more serious issues of identity and privacy as students approach their adolescent years.

There is some difference in what is taught around “algorithms”, which covers both the design of simple programs, as well as understanding algorithms for standard problems such as searching and sorting. These standard problems serve as examples of clearly defined problems, but also allow students to investigate their performance. The Australian curriculum starts earlier with standard problems, but by 11 years of age all three curricula include such algorithms. This will be another important area to evaluate in studies with students to determine whether it is worthwhile starting early with these concepts.

Webb et al. (2018) discussed the evidence that young student, of 7 or 8 years of age can start to develop understanding of important Informatics concepts. Students can learn through hands-on experience and gradually begin to link theoretical concepts to their developing practical problem-solving capabilities. Therefore, identifying trajectories in the development of

these concepts and devising effective pedagogical approaches which make use of the tools available are important current research challenges. Furthermore, in addition to developing Informatics concepts to support the subject per se, it is necessary to define the underlying knowledge base of Informatics concepts and crucial skills needed to support digital citizenship.

There are some suggestions about introducing Informatics in primary education in Poland: Informatics activities need to be included in the same place where kids are playing, so there is no need for a fully equipped classroom. Integration of Informatics with other subjects during the whole week (1 hour lasts a week). Sometimes a teacher may take students to a computer laboratory. Teachers have access to students' results regardless of the place they work, at school or at home (homework). The flipped learning method is suggested to be used (Sysło, 2017).

While computational thinking is just one element of Informatics, Angeli et al. (2016) suggested designing a curriculum for primary school with an explicit focus on computational thinking, before covering more theoretical and applied concepts of Informatics in secondary education.

Six core learning areas have been announced in the curriculum of New Zealand: (1) algorithms, (2) programming, (3) data representation, (4) digital devices and infrastructure, (5) digital applications, and (6) humans and computers. The suggestion that these areas should be related to the principles of computational thinking is made (Duncan, Bell, Atlas, 2017).

Angeli et al., (2016) support the holistic design approach for teaching computational thinking and emphasize two steps: (a) the design of problem solving tasks with a focus on real-life issues, and (b) the sequencing of problem solving tasks from simple to complex. It is also evident that children may need guidance and support as they start working on more challenging tasks. Support may come from the teachers, but for them it is also important to have pedagogical content knowledge, in order to better explain what students need to know.

There are many ways for selecting problems to be solved by students in the classroom. For primary education two types of problem solving are usually declared:

1. Practical problems which take more time and cover several topics;
2. Everyday exercises (they are very common in mathematics and language [grammar] lessons).

In this thesis we provide the third type of problems - short tasks with a double-sided aim: to cover Informatics concepts and to be solvable in a few minutes (more in the next subsection).

The conclusion is that many countries are integrating digital competencies in primary education already and introducing basics of Informatics by using various activities.

2.5 Case of Computational Thinking Activity

Attracting youngsters to choose Informatics at school has always been a challenge for educators. Understanding and handling the basics and foundations of Informatics or computing is more important than knowing many technical details.

For this purpose, the idea of developing a contest on Informatics fundamentals for school students was raised by Lithuania in 2004 (Dagiene, 2005; Dagiene, 2006). The Bebras contest (www.bebras.org) focuses on understanding Informatics concepts and phenomena. In 2015, the Bebras contest on Informatics and computer fluency was renamed the Bebras contest on Informatics and computational thinking. Nowadays it is based on the expression of Informatics concepts in attractive, interesting, and fun tasks. Specifically, the idea is to encourage children to learn Informatics fundamentals (concepts), and to support the development of algorithmic thinking as well as computational thinking (Dagienė, Stupurienė, 2014; Dagienė et al., 2014).

From a single contest-focused annual event Bebras has developed into a multifunctional contest and an activities-based educational model. The model combines both international and national levels and involves a variety of activities, especially at the country level (Stupurienė et al., 2016). Recently, this contest has been spreading to 68 countries (2019 April data) all around the world: Australia, Austria, Algeria, Azerbaijan, Belarus, Belgium, Bulgaria, Bosnia and Herzegovina, Brazil, Cambodia, Canada, China, Croatia, Cyprus, Czech Republic, Dominican Republic, Egypt, El Salvador, Estonia, Finland, France, Germany, Greece, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Ireland, Israel, Italy, Japan, Jordan, Kazakhstan, Latvia, Lithuania, Malaysia, Malta, Mongolia, Netherlands, New Zealand, Nigeria, North Macedonia, Norway, State of Palestine, Pakistan, Poland, Portugal, Romania, Russian Federation, Serbia, Singapore, Slovakia, Slovenia, South Africa, South Korea, Spain, Sweden, Switzerland, Taiwan, Thailand, Tunisia, Turkey, UK, Ukraine, USA, Vietnam.

Countries involve various activities, e.g. several rounds of the challenge, discussion on Informatics topics, task solving seminars, teacher workshops, and task developing events.

For better understanding in which direction the research concerning the Bebras contest is going, Dagienė and Stupurienė (2016a) decided to make a revision of research publications by using a systematic literature review.

Sources: Web of Science, ACM Digital Library, Springer Link, SCOPUS, Google Scholar. Search term: Beaver contest OR Bebras contest OR Bebras challenge. Language: English. Bebras was established in 2004, and the first publication appeared in 2005, so the time range was 2005–2015.

We found 149 papers, but only 76 met the language criteria and were selected for further analysis. Remaining 73 papers were published in different languages (Czech, Finnish, French, German, Italian, Japanese, Lithuanian, Russian, Slovakian, Slovenian, and Mandarin Chinese). The papers are divided into three categories: 1. The Bebras contest is the main source for research question (39 papers); 2. The Bebras contest is discussed as a good practice example for Informatics education (17 papers); 3. The Bebras contest is only mentioned between other activities (20 papers). The number of research papers has significantly grown during the last years; Fig. 9 represents the dynamic of publications on the topic.

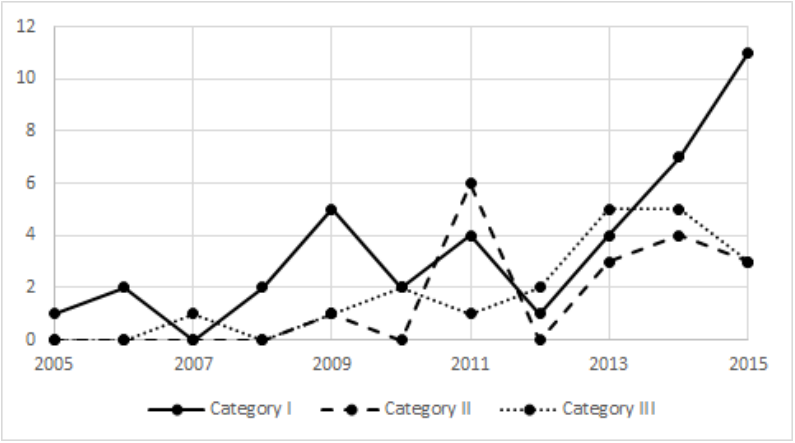


Fig. 9. Number of papers by years and categories

At first, all the 76 papers were analyzed by their keywords in order to find out the key topics of the Bebras contest. Naturally, Informatics education, Computer Science, programming, contest in learning, learning, computational thinking, and problem solving are the most dominating topics (Fig. 10).

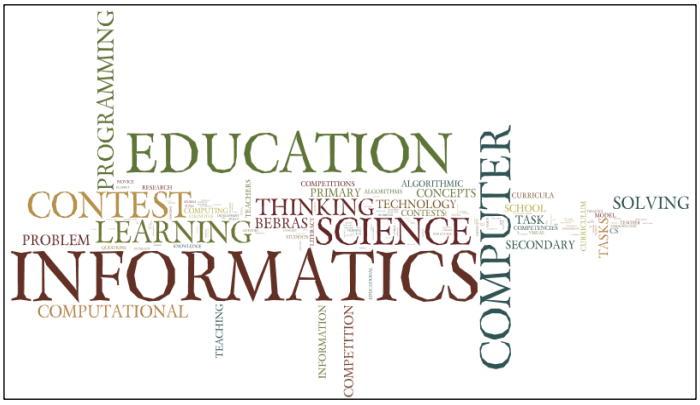


Fig. 10. Cloud of keywords found in publications

Further the authors analyzed 56 papers where the Bebras contest was the main source of research questions or served as reasoning of good Informatics education practices.

The review was driven by the following research questions:

1. What evidence is there in the literature to identify that the Bebras contest is effective enough in promoting Informatics concepts and computational thinking in children and young people?
2. What evidence is there in the literature how to develop the quality of the Bebras contest?
3. What evidence is there in the literature to see the impact of the Bebras contest on formal and/or non-formal Informatics education in countries?

A more detailed overview is provided in the paper by Dagienė and Stupurienė (2016b).

The worldwide Bebras Informatics contest is discussed as an example of connecting formal and non-formal Informatics education by using thousands of tasks based on Informatics concepts and applying problem-solving strategies (Dagienė, 2018).

It appeared that tasks are the core elements of the Bebras model. The quality of tasks is crucial for the success of the Bebras contest. Tasks are an important source for introducing kids to Informatics concepts and procedures.

Bebras tasks are short, answerable in a few minutes through a computerized interface or in some cases even with pen and paper, and require deep-thinking skills in the Informatics field and no pre-knowledge is required. To solve those tasks, students are required to think in and about

information, discrete structures, computation, data processing, data visualization, and they should use algorithmic as well as programming concepts. Each Bebras task has a dual aim: to demonstrate an aspect of Informatics and to test the participant's ability to understand Informatics fundamentals (Dagiene et al., 2015a; Dagiene et al., 2015b).

The contest should help children to get interested in Informatics and to stimulate thinking about contributions of Informatics to science at the very beginning of school (Dagiene, 2010). The Bebras contest may play an important role in creating the school curricula from the "bottom", from basic elements and individual questions upon which broader Informatics concepts may be introduced (Vaníček, 2013; Vaníček, 2014).

As suggested by Dagienė and Stupurienė (2016a), an Informatics learning task developing process (spiral cycle) (Fig. 11) begins with a chosen Informatics concept, which is the key idea what we want to teach the students. Usually a text with the visual components is created by involving in a story or fiction. By using gamification (application of game principles in non-game contexts) and by adding dynamic components (dragging, dropping, etc.) a task for the Bebras contest can be created. A Bebras task is usually modified several times (using the iterative method): simplified in text, better explained and presented or changed in its story or the question is changed and sometimes even the type of task is changed as well (dynamic, multiple choice, open-ended).

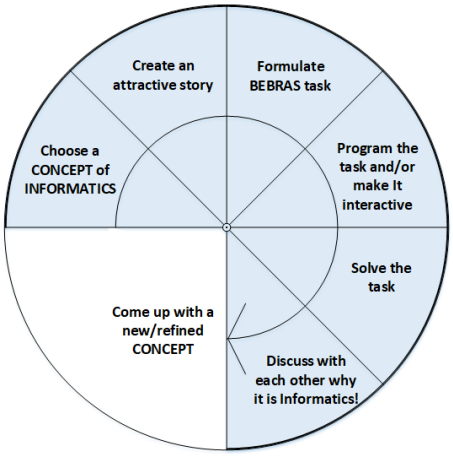


Fig. 11. The task developing process (Dagienė, Stupurienė, 2016a)

The Bebras tasks code and design process by using a tool is presented in the paper published by Dagienė, V., Stupurienė, G., Vinikienė, L. (2017b).

The creation of tasks for learning Informatics concepts is a constructive way of learning. Teachers have the freedom to create any task that is useful for the Bebras contest. In creating tasks, a deconstructionist way of learning also takes place: Informatics concept is analyzed and deconstructed in its main aspects; some of these aspects are chosen for the task creation where these aspects are constructed to a suitable task. Very often a suitable story has to be invented that enables us to convey the aspects of the Informatics concept in an easy way. Creating (constructing) tasks have the same importance as deconstructing the given task – to find out what concepts are hidden in the task and to provide a conceptual bridge to Informatics science.

The most important goal of the Bebras contest is to present Informatics concepts in an understandable way and an attractive format so that everybody could learn these concepts and would be motivated to learn Informatics further on.

2.6 Template of ICDT

The development of learning tasks for Informatics contest is important: they must cover Informatics concepts and as many areas of the discipline as possible. Moreover, the tasks have to be selected carefully, with regard to the different aspects of each task (i.e., how the topic is pitched) and evaluation of its attractiveness to students (whether it stimulates learning and discovery).

Based on the previous rich experience collected by the international community of the Bebras contest in creating learning tasks, a set of requirements was formulated (with the participation of the author). Every participating country may decide which attributes can be included in their tasks. However, the following requirements are mandatory for the international version of task (Fig. 12).

First, a task must have **a title**. This title is displayed to the students during the contest; it may change over time and differ from translation to translation.

Age groups depend on the biological age of students and are defined as follows: I group: 6-8 years of age; II group: 8-10 years of age; III group: 10-12 years of age; IV group: 12-14 years of age; V group: 14-16 years of age; VI group: 16-19 years of age.

Difficulty is a measure of complexity of the task for students in a particular group: easy, medium, or hard.

2019-XY-01-eng Task Name

I: ----	II: ----	III: ----	IV: ----	V: ----	VI: ----
Category:	☐ ALP	☐ DSR	☐ CPH	☐ COM	☐ ISS
Keywords:					

Answer Type: Click to choose!

Task Title (may be different from the task name)

Insert the task text and images here!

Question / Challenge

In case of a multiple-choice or open-answer task, insert the question here.
In case of a constructive task, insert the challenge here.

In case of a constructive task, instructions for the interaction are given here, if needed.

Answers

In case of a multiple-choice task: Specify the answer choices here!

A) Answer 1
B) Answer 2
C) Answer 3
D) Answer 4

In case of an open-answer task: Specify the range of answers that an implementation of a task should accept! (Examples: Integer numbers from [0,99]; strings of 4 capital letters; …)

In case of a constructive task: Specify what the task script will accept as contestant input.

The correct answer is:

Explain which is the correct answer and why. In case of a multiple-choice task, also explain why the other answer choices are incorrect. You may explain and motivate your choice of wrong answers in the comment section below.

If the task asks for an optimum, you should be able to prove the optimality of the correct answer.

Students (Bebras participants of the relevant age) must be able to understand this explanation. Use about 2 to 5 sentences for the correct answer, and somewhat less for the wrong answers. Focus narrowly on the task, do not explain yet what this has to do with informatics.

It's informatics!

Explain **to the target age group**, why this task is about informatics (and computational thinking): What are the informatics concepts, what is the informatics "story" behind this task? Use about 3 to 8 sentences. **Do not explain the correct answers of a task**, but give a larger picture.

If there are several concepts in this task, it is recommended to focus on one of them. It might be nice to also add one or two relevant web-links here, for further reading. This text is for teachers and students at the same time.

Wording

List of words **and phrases** used to name important things mentioned in the task body (actors, activities/processes, concepts, definitions, objects, names, etc.)

Comments

Important remarks, like: explanation of a significant modification to the task, suggestion of potential variants of the task, … Please try to stick to the following format: 2019-05-08 Firstname Lastname (Country), address@provider.com

Graphics

For each image, please name the author and/or the source. For images from third parties, please explain why the image is free to be used in the task. Note: If the source of an image is unclear, it **might be copyrighted and cannot be used in the task**. A simple example:

Image 1: self-made by Firstname Lastname (Country), address@provider.com

(continues in next page)

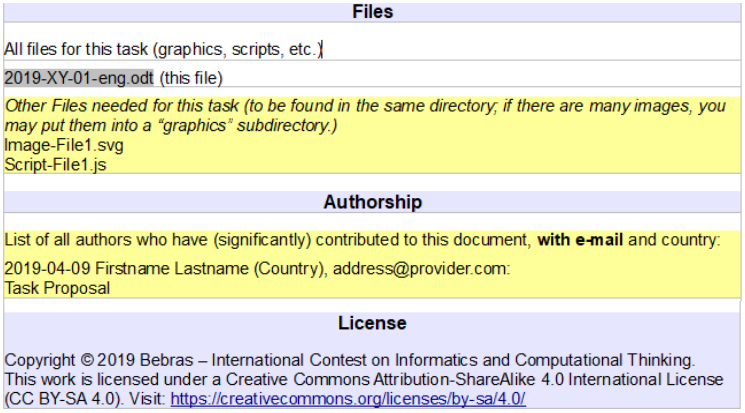


Fig. 12. Structure (framework) of Informatics learning task

In general, it is important to develop a categorization system or taxonomy of learning tasks. Based on previous learning tasks' category systems for Bebras tasks (Opmanis et al., 2006; Dagiene, Futschek, 2008), new categorization system was proposed in 2016 by Dagiene, Sentance and with direct involvement of the author, and used from 2017. The content of school Informatics can be divided into five knowledge areas (content categories):

1. Algorithms and programming, including logical reasoning (ALP);
2. Data, data structures and representations (includes graphs, automaton, data mining) (DSR);
3. Computer processes and hardware (includes anything to do with how the computer works – scheduling, parallel processing) (CPH);
4. Communications and networking (includes cryptography, cloud computing) (COM);
5. Interaction (Human-Computer Interaction, HCI), systems and society (all other topics!) (ISS).

These Informatics areas are used for the **Category** attribute in task from 2017.

For practical use, when developing or using Informatics tasks, a precise description of each category is needed. One way of achieving this is the use of keywords. **Keywords** are important to assist in the categorization. They will also be important to teachers who wish to find tasks that fit with the topic being taught in the curriculum (Dagiene, Sentance, 2016; Yang, Park, 2014). Therefore, keywords information should be retained with the task to help Bebras users select from previous tasks and identify teaching topics around Bebras tasks. Practically no more than three keywords are necessary.

There are three **types** of answers to tasks and it depends on contest management systems features. Types: (1) multiple-choice (text / image); (2) open input (integer / text); (3) constructive (script-based)

The **Task body** consists of task text, images (optional) and question. In case of a multiple-choice or open-answer task, it is necessary to insert the question. In case of a constructive task, the contest and instructions for the interaction should be inserted, if needed.

Answer. In case of a multiple-choice task there should be four possible answers. In case of an open-answer task it is needed to specify the range of answers that an implementation of a task should accept, for examples: integer numbers from [0,99]; strings of four capital letters. In case of a constructive task what the task script will accept as contestant input has to be specified.

The correct answer is an explanation which is the correct answer and why. In case of a multiple-choice task, it should also be explained why the other answer choices are incorrect and motivate the choice of wrong answers in the **comment** section below. If the task asks for an optimum, you should be able to prove the optimality of the correct answer. Also students (Bebras participants of the relevant age) must be able to understand this explanation. Focus narrowly on the task; do not explain yet what this has to do with Informatics.

In "It's Informatics!" part there is an explanation to the target age group, why this task is about Informatics (and computational thinking): What are the Informatics concepts, what is the Informatics "story" behind this task? Do not explain the correct answers of a task, but give a larger picture. If there are several concepts in this task, it is recommended to focus on one of them. It might be also nice to add one or two relevant web-links here, for further reading. This text is both for teachers and students.

Note that not all parts of the task are shown to the students during the solving time in the information system. Some of the parts are necessary for the teachers, for example, explanations. As many of our teachers have no formal training in Informatics, it is very useful for them to understand why answers are correct or incorrect so as to form their future teaching activities. Other parts of the task in this thesis are not described in detail.

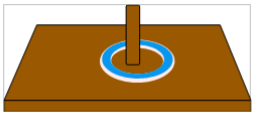
An example of a learning task is presented in Fig. 13. This task is created by representatives from Malaysia, who are members of the international Bebras community. The version presented is the primary source and will be used for translation to other languages and for implementation in various contest management systems.

2018-MY-08-Rubber Band

I easy	II: easy	III: ----	IV: ----	V: ----	VI: ----
Category:	☐ ALP	☑ DSR	☐ CPH	☐ COM	☐ ISS
Keywords: Sequence					
Answer Type: Open Integer					

Ring Toss

Sarah plays ring toss with her friends. Each of them takes turns to toss five rings around a peg:

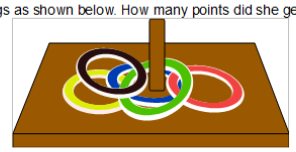


An incorrect toss wins no point. Every correct toss wins a point:

Toss	points
first toss	5
second toss	4
third toss	3
fourth toss	2
fifth toss	1

Question / Challenge

Sarah tossed her five rings as shown below. How many points did she get?



Answers

Write your answer in the box below.

The correct answer is:

Sarah won 6 points.

Toss	ring	✓ / ×	points
first toss	yellow	×	0
second toss	blue	✓	4
third toss	pink	×	0
fourth toss	green	✓	2
fifth toss	black	×	0
TOTAL			6

It's informatics!

The management of data structure, namely a stack, is shown. This problem is a sequencing problem. A computer analyses data in sequence. Data needs to be organised so it can be processed to help determine solutions.

Fig. 13. An example of the Informatics learning task

2.7 Two-Dimensional Categorization

Conceptualization is formation of concepts (Papaurelytė-Klovienė, 2007) (see subsection 2.1). George Lakoff (1987) in his famous book "Women, Fire and Dangerous Things" states that there is nothing more important than categorization of our thought, perception, action and speech. Whenever we think about something, we are categorizing. Things surrounding us are categorized and grouped together according to what they have in common.

When we deal with concepts, we cannot forget the importance of conceptualization and categorization. The process of conceptualization allows us to form concepts in our minds. Categorization allows us to categorize them according to some common features.

According to Jacob (2004), categorization is the process of dividing the world into groups of entities whose members are in some way similar to each other. Categorization is the basic cognitive process of arranging objects into categories. It is a fundamental process in human and machine intelligence and is probably central to investigations and research in cognitive science (Cohen, Lefebvre, 2005). It is important to develop a categorization system or taxonomy of learning tasks.

A new categorization system for learning tasks that includes both content areas of Informatics (knowledge) and computational thinking (skills) was proposed by Dagiene, Sentance and Stupuriene in 2016. The main reasons were: (1) Categorization can help keep track of what type of tasks are being used; (2) Can help identify particular tasks for use in the curriculum; (3) Can help task developers to write tasks around varied areas of the curriculum; (4) To ensure a balance of tasks across a range of Informatics concepts.

Computational thinking is an increasingly important focus within Informatics curricula around the world and ways of incorporating it into the school curricula are being sought (Dolgopolovas et al., 2015).

The area of computational thinking covers a range of different skills relating to problem-solving. The issue becomes the need to select a categorization system which is true to the definition of computational thinking whilst encompassing the range of skills that students utilize when solving learning tasks. There are two advantages of incorporating this into the revised category system: (1) Task development can focus more closely on how computational thinking skills are being developed or utilized; (2) Teachers and students can relate the learning from the task to their understanding of computational thinking when the tasks are discussed during the lessons.

For practical use, when developing learning tasks, a precise description of each category is needed. One way of achieving this uses keywords.

Each category (content domain) and keywords were discussed with researchers and experts from Informatics education (mostly from the international Bebras community) and comparing previously used categorization systems (Opmanis et al., 2006; Dagiene, Futschek, 2008; Kalas, Tomcsanyiova, 2009). A suggested set of keywords is shown in Table 4. Keywords are important for assisting the categorization of the tasks.

Table 4. Informatics content domains and keywords

Domain	**Keywords**
Algorithms and programming	Algorithm; Binary search; Boolean algebra; Breadth-first search; Brute-force search; Bubble sort; Coding; Computational complexity; Constants; Constraints; Debugging; Depth-first search; Dijkstra's algorithm; Dynamic programming; Divide and conquer; Encapsulation; Function; Greedy algorithm; Heuristic; IF conditions; Inheritance; Iteration; Kruskal's algorithm; Logic gates; Loop; Maximum flow problem; Objects; Operations AND, OR, NOT; Optimization; Parameters; Prim's algorithm; Procedure; Program; Programming language; Program execution; Quick sort; Recursion; RSA algorithm; Shortest path; Searching; Sorting; Traveling salesman problem; Variables.
Data, data structures and representations	Array; Attributes; Biconnected graph; Binary and hexadecimal representations; Binary tree; Character encoding; Databases; Data mining; Eulerian path; Finite-state machine; Flowcharts; Fractals; Graph; Hash table; Integer; Information; Linked list; List; Queue; Record; Stack; String.
Computer processes and hardware	Cloud computing; Deadlock; Fetch-execute cycle; Grid computing; Image processing; Interpreter; Memory; Multithreading; Operating systems; Parallel processing; Peripherals; Priorities; RAID array; Registers; Scheduling; Sound processing; Translator; Turing machine.
Communication and networking	Client/server; Computer networks; Cryptography; Cryptology; E-commerce; Encryption; Parity bit; Protocols; Security; Topologies.
Interactions, systems and society	Classification; Computer use; Design; Ethics; Graphical user interface; Interaction; Legal issues; Robotics; Social issues, Virus.

A suggested categorization of computational thinking skills follows the work of Selby and Woollard (2013), which has been adopted by Computing at School in the UK in developing guidance on computational thinking for

teachers (Csizmadia et al., 2015). This describes aspects of computational thinking skills exhibited by learners as falling into the five categories below:

1. Abstraction;
2. Algorithmic thinking;
3. Decomposition;
4. Evaluation;
5. Generalization.

The use of keywords will be slightly different for computational thinking skills. Classifiers need to know how to identify whether that skill can be used to solve that task (Table 5). One of the difficulties is that we can only presume how the learner solves the task which may be a different way to the way the task setter might solve the task. This means that more than one computational thinking skill may be associated with each task. We are suggesting a maximum of three, in order to concentrate more on understanding of them.

Table 5. Computational thinking skills and ways to identify them

Computational thinking skill	**How to spot the use of that skill**
Abstraction	Removing unnecessary details; Spotting key elements in problem; Choosing a representation of a system.
Algorithmic thinking	Thinking in terms of sequences and rules; Executing an algorithm; Creating an algorithm.
Decomposition	Breaking down tasks; Thinking about problems in terms of component parts; Making decisions about dividing into sub-tasks with integration in mind, e.g. deduction.
Evaluation	Finding best solution; Making decisions about whether good use of resources; Fitness for purpose.
Generalization	Identifying patterns as well as similarities and connections; Solving new problems based on already-solved problems; Utilizing the general solution, e.g. induction.

Incorporating both described categorization systems we can compose a two-dimensional system which can be represented as shown in

Table 6.

Table 6. Two-Dimensional categorization system

	Algorithms and programming	Data, data structures and representations	Computer processes and hardware	Communication and networking	Interactions, systems and society
Abstraction					
Algorithmic thinking					
Decomposition					
Evaluation					
Generalisation					

The suggested categorization system incorporates both computational thinking skills and Informatics concepts in the classification of learning tasks.

The presentation of this schema as a 2-D matrix merely indicates that every computational thinking skill can occur with each of the concept ideas – there is no dependency between the two classifiers. In practical terms, a task should be allocated to one Informatics content domain only but may have up to three computational thinking skills identified. Computational thinking skills are more difficult to clearly define and identify in a task as they are dependent on the approach taken to solve the problem; thus some flexibility is needed here.

The categorization system could be used in addition to encourage the development of tasks that use a variety of Informatics topic areas as well as computational thinking skills. On the other hand, this system helps Informatics teachers to choose the content of a lesson and provides them with a tool effective to select the tasks according to the particular topic.

The matrix presented in Table 7 demonstrates that this schema can be seen as a two-dimensional one. In practical terms, a template has been designed for developers to assign categories to tasks, including keywords (Table 4).

Table 7. A template table for task categorization

Name of task	Informatics domain	Keywords (≤3)	CT Skill (≤3)

First, this approach is quite complex. It gives more finely-grained classification that will produce much more useful outputs as a number of available tasks for teaching purposes. However, a more finely-grained system requires more knowledge and understanding of how to implement it

correctly. Task developers in different countries may not be able (or not willing) to assign the level of the detailed categorization of each task.

Second, not all teachers can be familiar with computational thinking, and understanding of the component skills presented here may not be shared. So teachers will need clear examples of computational thinking skills in learning tasks and explanations should be available to ensure some consistency of allocation of computational thinking skills to task.

Third, related to this, we will need to develop more precision in allocating computational thinking skills to tasks. The description by Wing (2006) that "computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to Computer Science" may lead us to think that computational thinking is everywhere and the composite skills appear in all tasks. A liberal interpretation such as this may render the computational thinking skill allocation to be meaningless. Computational thinking skills should only be allocated to a task where there is some element of Informatics in the task that develops this skill. For the future implementation it would necessary not only provide the list of CT, but also how much of each skill is expressed in the task (could be in scale from 1 to 10, or in percentages).

With due attention to the points raised above, the purpose of this development is to build up a bank of tasks which are categorized using the proposed framework. This will enable teachers to find useful tasks that they can use in the curriculum. It will also help task developers to focus on writing tasks around topics that are under-represented in the bank of tasks. An online search facility could be implemented to assist teachers looking for tasks on certain topics via keywords, concepts or computational thinking skills.

Tasks are very important both for students and task developers (teachers): students should be encouraged to think about Informatics, educators should think about the harmonization of the syllabus of Informatics.

The evaluation of the proposed categorization system is provided in Section 4.

2.7.1 Examples of ICDT

In order to illustrate the two-dimensional categorization system, we will describe here four examples of Informatics concept-driven tasks created by the international Bebras community.

Example 1. The task title is **Strawberry hunt.** Age group: grades 1 and 2. Difficulty - medium. Informatics domain - Data, data structures and

representations. Keywords - Graph, Edges, Nodes. Computational thinking skills - Abstraction, Algorithmic Thinking, Evaluation. Authorship: Dasović Rakijašić (Croatia)

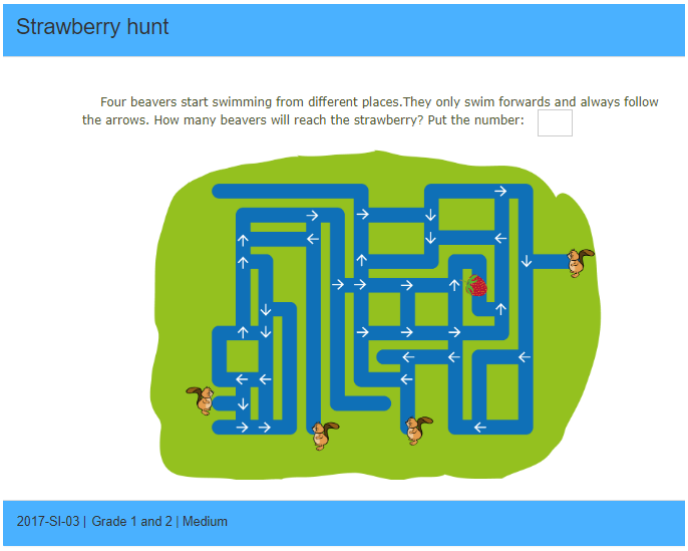


Fig. 14. ICDT title: Strawberry hunt

Explanation

The system of the canals in which the beavers are swimming has two main elements: canals (where the beavers can swim through) and crossings (where the beavers have to decide, by the arrow, into which canal to swim next). In Computer Science this system is called a graph with edges (the canals) and nodes (the crossings). In this case the nodes have extra information attached to them: which canal should the beaver swim into next.

"It's Informatics!"

Graphs can be used to describe situations like this task. They can also be used for programming a computer: the computer is following a path in the graph and at each crossing it receives an instruction on what to do next. In some cases, it ends up solving the problem (which would be the beaver reaching the strawberry) and in some cases it ends up in a dead end or even never finishes the program (like the two other beavers).

Example 2. The task title is **Sticks and shields**. Age group: grades 3 and 4. Difficulty - hard. Informatics domain - Algorithms and Programming. Keyword - searching, backtracking, pruning. Computational thinking skills - Algorithmic Thinking, Decomposition, Evaluation. Authorship: Hiroki Manabe (Japan), Momo Yokoyama (Japan), Maiko Shimabuku (Japan).

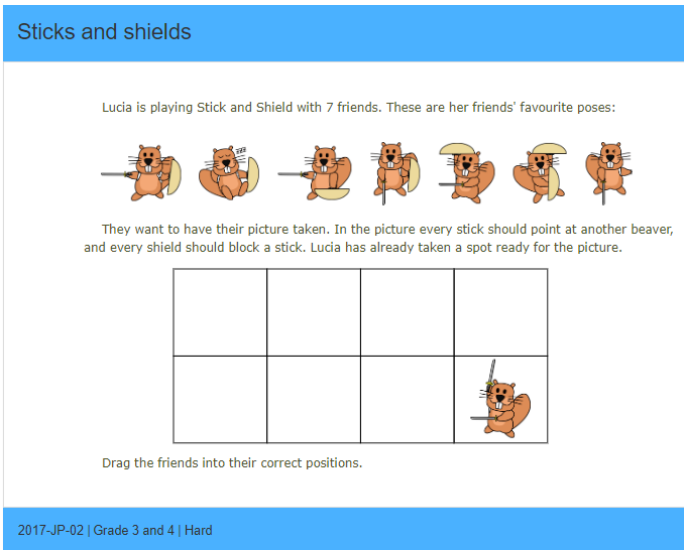
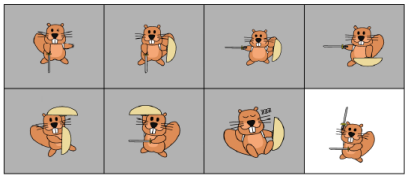


Fig. 15. ICDT title: Sticks and shields

Explanation

This is a task where the solution has to satisfy particular criteria. It is also a task where the number of possible arrangements is quite high but not many are correct. The first thing to do when solving this problem is to split the beavers into those that have to be on the top row, those that have to be on the bottom row and those that can be anywhere. This simplifies the task somewhat; however, it is still not an easy problem!



"It's Informatics!"

This could actually be a very complicated puzzle. Just a few pictures lead to a very time-consuming search among all possible (but incorrect) solutions. If you add just one more picture to a puzzle of six pieces, you would have six times as many different possibilities of placing the seven cards in the empty spots. For n cards, you have (n-1)! = 1 x 2 x 3 x … x (n-2) x (n-1) different possible solutions. So in this case there are 720 different possible solutions (but almost all of them are wrong).

However, using some logical thinking the search space can be pruned a lot. For instance, all beavers with a stick pointing down must be placed on

the top row, and there is only a single beaver that can be placed right above Lucia. A full exhaustive search can be done using an algorithm called backtracking. Using the backtracking algorithm, the search space can get really large. This is why pruning is important.

Example 3. The task title is **Parking lot.** Age group: grades 3 and 4. Difficulty - medium. Informatics domain - Data, data structures and representations. Keywords - Bit, Binary Code, OR logical operation. Computational thinking skills - Algorithmic Thinking, Decomposition, Evaluation. Authorship: J.P. Pretti (Canada).

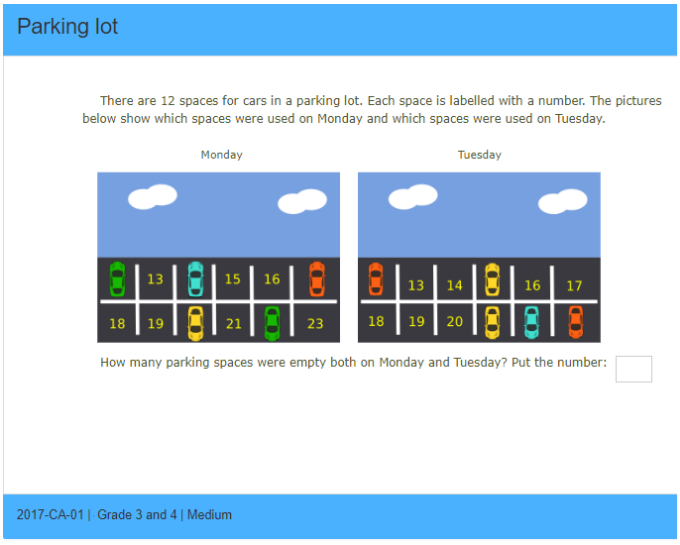
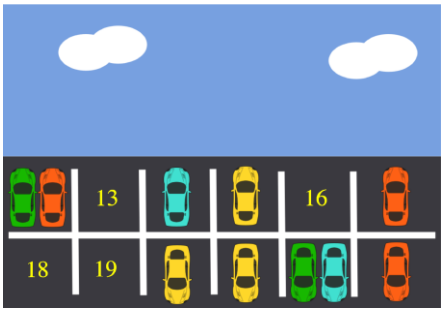


Fig. 16. ICDT title: Parking lot

Explanation

The answer is four spaces. Placing the pictures of the cars from both days together in the parking spaces, gives the image on the right. Then all we have to do is count the empty spaces.



“It's Informatics!”

All data can be thought of as a sequence of zeros and ones. Each zero or one is called a “bit” and the sequence is called a binary code, binary representation, or binary number.

Here, we can model the presence of a car as a “one” and an empty parking space as a “zero”; so the parking space corresponds to a “bit”. We get a sequence of bits if we view the parking spaces in order.

For example, we might move across the top row and then along the bottom row to get 101001001010 from the parking lot on Monday and 100100000111 from the parking lot on Tuesday. This task tells you to determine which of the twelve positions contain a 1 in either of these binary numbers. This is an operation named OR. Notice how we can compute the correct answer by seeing that 101001001010 OR 100100000111 gives 10110100111. This resulting binary number has four zeros in it.

Example 4. The task title is **The way home.** Age group: grades 3 and 4. Difficulty - medium. Informatics domain - Algorithms and programming. Keywords - Route, Backward searching, Black holes. Computational thinking skills - Algorithmic Thinking, Decomposition, Evaluation. Authorship: Zhukovsky Serhij (Ukraine).

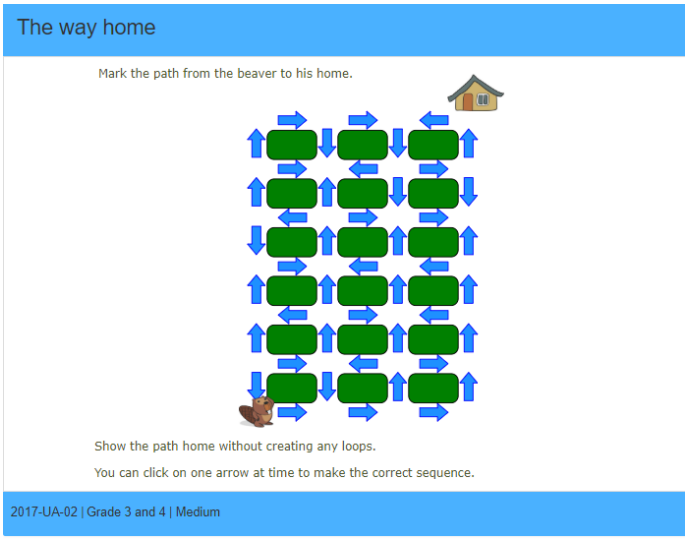
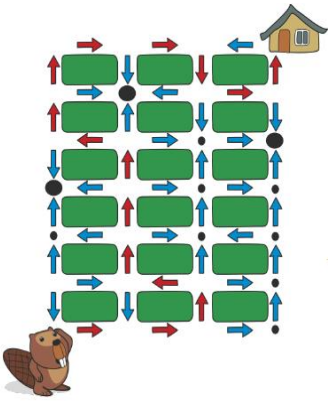


Fig. 17. ICDT title: The way home

Explanation

One way of solving this is to first identify black holes (see big black dots on the left) where the beaver can enter but not escape. We can also identify places that can only lead to a black hole (little black dots). The answer then

becomes obvious. An alternative strategy is to follow the arrows backwards from the house.



"It's Informatics!"

Finding the route is one of the classical problems in algorithm theory. Backward searching and identifying black holes are two algorithmic techniques used to solve such problems.

2.7.2 Analysis of Existing ICDT

In this subsection statistical analysis of Informatics concept-driven tasks created during a four-year period (2015-2018) by the community of the international contest on Informatics and computational thinking (see more about this contest in Section 2.5) is described.

The development of tasks for an educational contest is very important: they should cover fundamentals and as many subareas of discipline as possible. Moreover, the tasks have to be selected carefully, with regard to the different aspects of each task (i.e. how the topic is pitched) and interpretation of its attractiveness to students (whether it stimulates learning and discovery).

International Bebras task developing workshops have been organized annually (since 2005) and they bring together the representatives of all these countries for hard work and making decisions on good tasks for promoting Informatics education at primary and secondary schools. The tasks are created by representatives from various countries within the Bebras community. Each participating country provides from five to ten ICDT. Any member of the Bebras community may act as a reviewer. At least two reviewers are assigned for each proposed task to make a comments and rate

the tasks. It is called pre-workshop review process and review platform is developed in order to accelerate the reviewing process.

After tasks reviewing process finishes, discussions on accepted tasks begins in annual workshop. A result of that is the list of accepted tasks for the contest. Each task is further developed after the Bebras workshop within the community and there are different forms of tasks in various countries.

Every year a large amount of tasks is created by the Bebras community. In this thesis the proposed and accepted tasks during the period of 2015-2018 are analyzed. During the four years a total of 848 Informatics tasks were analyzed (Fig. 18).

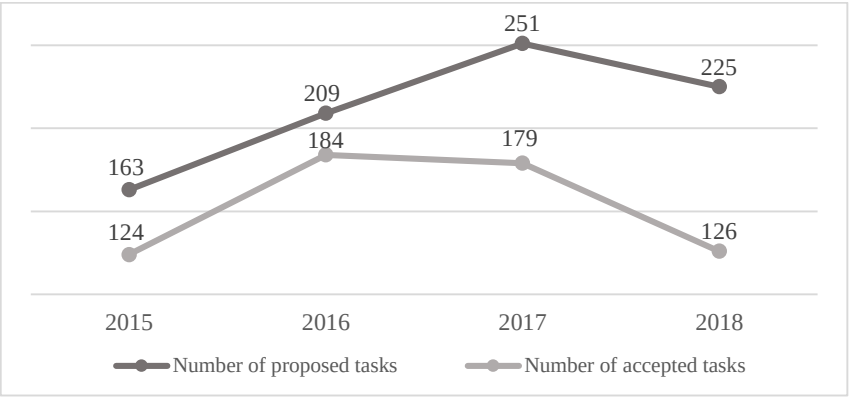


Fig. 18. Proposed and accepted tasks from 2015-2018

Categorization of the tasks is a significant point and it ensures that tasks span a wide range of topics. The categories proposed by Dagiene and Futschek (2008) were used between 2008 and 2016.

The categories proposed by Dagiene and Futschek (2008) were used between 2008 and 2016:

1) Information comprehension (INF);
2) Algorithmic thinking (ALG);
3) Structures, patterns and arrangements (STRUC);
4) Puzzles (logical) (PUZ);
5) Using computer systems (USE);
6) Social, ethical, cultural, international, and legal issues (SOC).

As mentioned in the previous section (see Section 2.6), a new categorization system was proposed in 2016 by Dagiene, Sentance, Stupuriene and it was used from 2017 year. It is based on a two-dimensional approach: integrates Informatics concepts together with computational thinking skills.

A content analysis of 613 accepted categorized tasks was performed in respect of tasks categorization systems. The old categorization system was used for 2015 and 2016 years' tasks. The new tasks' categorization system was used for 2017 and 2018. In order to compare the results, were match two different categorization systems were matched and they are not very accurate, but the main topics are covered. The results are presented in Table 8.

Table 8. Matching of categorization systems

Old system (2008–2016)	**New system (from 2017)**
INF	DSR+COM
ALG	ALP
STRUC	DSR
SOC	ISS
USE	-
-	CPH
PUZ	-

It is important to mention that some of the tasks suit not one, but two or more different categories. The most popular categories and combination of their combinations (if accepted more than 10 tasks) in the old categorization systems are presented in Fig. 19. With the rules the same as for the old categorization system, the most popular categories for the new categorization systems are presented in

Fig. 20.

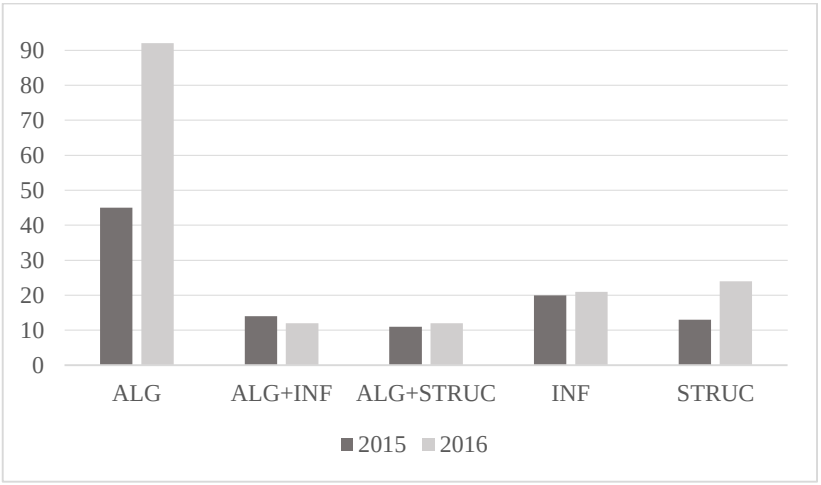


Fig. 19. Distribution of accepted tasks according to old categories (2015–2016)

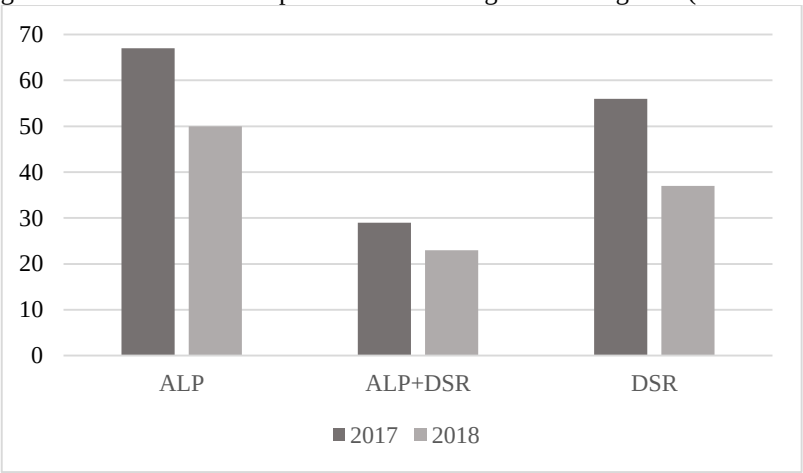


Fig. 20. Distribution of accepted tasks according to new categories (2017–2018)

As we can see from Table 9, the percentage of accepted tasks from the most popular categories is almost the same every year. It means (not directly) that with learning tasks developed by the community of the international contest, the following topics can be covered: algorithms, programming, logical reasoning, data, data structures and representations (includes graphs, automaton, and data mining).

Table 9. Percentage of accepted tasks the most popular categories

Year	2015	2016	2017	2018
Percentage	83 %	88 %	85 %	87 %

In this content analysis we did not pay attention to the distribution of computational thinking categories because the data is not available on them during 2015–2016.

2.8 Contest Management Systems

The contest management system (CMS) is the essential software environment in running the contest effectively and efficiently. For running a contest, more than 19 different CMS have been maintained in participating countries. CMS should support simple tools, which enable tasks development, users' management, announcement area, records of solutions,

reports, and data storage. The requirements for CMS were formulated in the paper written by Dagienė, Stupurienė, Vinikienė (2017a).

The survey designed by the author was conducted to compare and analyze different CMS. The main aim of this survey was to gather information about the systems used, and to get an understanding of the differences of the basic contest management principles. A questionnaire with fourteen open questions was announced in May 2016 and was accessible until the end of February (2017). The aim of this questionnaire was to collect information about CMSs in different Bebras community countries, to understand the real situation about CMSs (what is common, what is different), and to elaborate valuable suggestions for others.

Thirty-two countries (out of a total of 39 countries running the contest hereupon at 2017) answered the questionnaires regarding the Bebras contest.

Table 10 shows the distribution of countries by responsibilities of CMS support: (1) organizers in countries themselves create and develop the system; (2) support of the system is trusted for the private company (https://www.eljakim.nl/project/beverwedstrijd/); (3) organizers use platforms available on the Internet.

Table 10. Countries distribution by responsibilities of CMS support (2017 data)

CMS	Countries
Developed by organizers	Belgium, Bulgaria, Cyprus, Estonia, Finland, France, Hungary, Indonesia, Italy Latvia, Lithuania, Macedonia, Russia, Slovakia, Slovenia, Spain, Taiwan, Ukraine
Developed by a company	Canada, Germany, Ireland, Japan, Netherlands, Romania, Singapore, Switzerland, USA
Platforms	Croatia (Moodle), Turkey (Moodle), Belarus (Yandex Contest)

Sweden and Finland collaborated in the development of the same system. Their system is implemented using Ruby and dynamic tasks are created using JavaScript. Serbia uses Slovenia's well-developed system (from 2013). They decided that a system has to be of high performance, scalable, and fault-tolerant (Kristan et al., 2014). To achieve that, a three-tier architecture consisting of a front-end layer, a business logic layer, and a distributed database back-end layer was applied.

The CMS of France (http://castor-informatique.fr/) is one of the most well-developed: in this system, an optimized front-end is used which reduces

the number of requests from clients to the server and consequently the load on a server. While the web workload is distributed, all the web servers access a single relational database. The sessions are implemented using Memcached technology, reducing the load on a database while still maintaining a single session across all the web servers. If the web servers become unavailable, competitors are provided with a coded message at the end of their competition. They can send this coded message to the organizers by e-mail and have their results entered into the system. To minimize the communication between the web server and the clients, results are only submitted at the end of the competition, with no backup in case a competitors' web browser crashes.

Turkey uses LMS Moodle (however, they do not have dynamic tasks). It is easy to use although the interface should be changed and there needed to have a special template necessary for the contest (Kalelioglu et al., 2015). Croatia also uses LMS Moodle. Teachers there take care of editing and publishing tasks, and managing participants with Moodle, since every student has their own access to the system.

Belarus uses the Yandex:Contest system. It is similar to the ACM-like contest system where it is possible to check test-like questions. Macedonia prefers to use a self-developed web-based system, developed with node:js. The main features of this system are multiple browsers and devices support, statistics collection, data backup, and it can also sustain the connection loss of a server or database.

The crucial point for CMSs is the number of people who participate at the same time. It partially depends on the settings of the system. However, small countries with fewer participants do not measure this. For example, for France (more than 470 000 participants) the maximum number of participants is 10 000. The French platform is designed to handle much more, if needed. For Belarus, Croatia, Lithuania, and Turkey, the maximum number of participants at the same time is 1,000, and for Bulgaria, Finland, and Italy it is 500 participants. Slovenia is able to connect the largest number of participants at the same time (more than 20 000). There is no limitation for contestants in Ukraine because they work offline, their answers are recorded as files, and are collected afterwards.

Using CMS options or additional analytics tools, organizers can provide data about their system, devices, technical details, the number of participants, and answer statistics for those participants. For example, France and Russia collect information about devices or browser versions through Google Analytics. Belarus has installed Yandex:Metrica.

Twenty countries gather information about gender and seven countries were not interested in the gender issue. In some cases, such as Italy and Singapore, the possibility of indicating gender is optional. All countries collect the number of participants by age group, but in some CMSs, such as Belarus, France, and Romania, participants are listed with their precise ages. Personal information, such as the name, surname, school, or language is collected in individual cases. It depends on the countries' attitude to privacy rules and data publication. Some countries privacy rules forbid using the student data for statistical research.

After comparing the data regarding to the possible main components of the different Bebras CMS, it seems that according to functionality, design of the typical Bebras CMS should be modular and consists at least of 6 modules (Fig. 21).

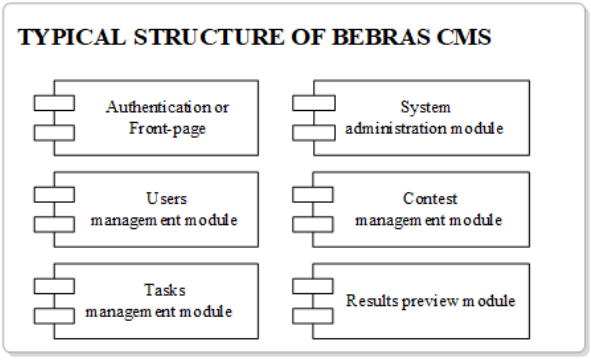


Fig. 21. The typical modular structure of the Bebras CMS

Experience with contest management inspires those interested to think about new system features or improvements for the present CMS. French Bebras organizers would like to have a tool for teachers for creating dynamic tasks (something similar to the Bebras Lodge tool; this is a special tool for creating and developing dynamic tasks), but with a different approach. Germany would like to have an API to import, store, and export Bebras tasks including their complete interactivity (the Bebras Pool). Lithuania has a plan to collect data regarding how many times participants have a second look at the same task or return to resolve that task, and how many times they have changed the answer. Slovakia wants to measure the time spent on each task. Ukraine would like to do compatibility with mobile devices and developer-friendly animations. Belarus, Canada, Croatia, Indonesia, Serbia and Turkey are planning to add different types of dynamic tasks or develop more

dynamic tasks' features. Macedonia would like to add more functionality to its CMS to create a friendlier environment for the teacher (for example, with ability to see information about students). Macedonia also emphasized the importance of testing as ensuring the performance of participants. Singapore is planning to introduce reports on results for schools and students into their CMS.

2.9 Summary

Embracing concept-driven Informatics education means that one needs to think about conceptual knowledge and deep learning, which can be encouraged by emphasizing principles and concepts rather than accumulated facts. Conceptual knowledge for a particular domain consists of the core concepts and their interrelations and can be characterized by using a number of different constructs, including semantic nets, hierarchies, and mental models. Learning theories, as constructionism, specify how individual learners construct mental models in order to understand the world around them.

The analysis of the existing Informatics education frameworks as well as their basic components and processes allows us making a conclusion that the most appropriate framework CDIE is cpm.4.CSE. The main reasons are that it clearly indicates the steps of the whole process and is closely related with determination of Informatics competences and concepts. However, the cpm.4.CSE should be improved/refined and modified for primary school context.

The systematic literature review was conducted to revise research publications in the CDIE field for better understanding in which direction the research concerning the Informatics concepts education is going. The analysis shows that there exist three types of concepts: (1) Informatics concepts; (2) computational thinking concepts; (3) programing concepts. The latter is often a part of Informatics concepts; therefore, it is not described as being separate concepts.

The analysis shows that students (like all humans) need motivation to learn things. One of the ways to encourage motivation is solving tasks. So we need to consider and design a new-task paradigm for future learning. Learning and understanding process of Informatics concepts will come later, actually after practice to solve many of concept-driven tasks. The teacher's role is important for strengthening the understanding of the Informatics

concepts. Teachers can help students to clarify tasks solutions, to explain why it is Informatics, and to provide resources for reading and discussions.

The template for developing the Informatics concept-driven task is presented based on 15 years of experience collected in creating and using such learning task by the community of international contests on Informatics and computational thinking. Analysis of tasks created by this community between 2015 and 2018 showed that in average 150 ICDT are accepted every year.

The two-dimensional categorization system for Informatics learning tasks has been developed. To make a classification of learning tasks, the categorization system incorporates both computational thinking skills and Informatics concepts.

Learning of Informatics concepts at an early age is important for a deeper understanding of various Informatics topics. Informatics concept-driven tasks focus on the concepts and support the understanding of Informatics phenomena. It is a promising way to develop computational thinking, which is probably one of the most important sets of skills for twenty-first century citizens.

The study of existing CMS has showed that there is a need to implement a structural selection of ICDT in the existing educational platform.

3. RESEARCH PART

In this section the process of development of the model for concept-driven Informatics education is presented. Separate parts of this model are introduced first in order to explain better the relations between them. The design of the educational platform extension is based on the developed model.

3.1 Extension of cpm.4.CSE Model

3.1.1 Functional Modeling Methodology

A modeling method comprises a specialized modeling language for representing a certain class of information, and modeling methodology for collecting, maintaining, and using the information so represented (Menzel, Mayer, 1998).

The methodology chosen for this research is the functional modeling family IDEFx. In particular, IDEF0 (Integration DEFinition level 0) is one of the widely spread techniques which is used as functional modeling methodology of all activities that affect the educational process (El-Sharef, El-Kilany, 2011).

As a business process model, IDEF is used to produce both descriptive and analytical models that support process development and design. The two primary modeling components used in IDEF0 are (IEEE Standard…, 1998):

1. Functions (represented by boxes on a diagram).
2. Data and objects that interrelate those functions (represented by arrows).

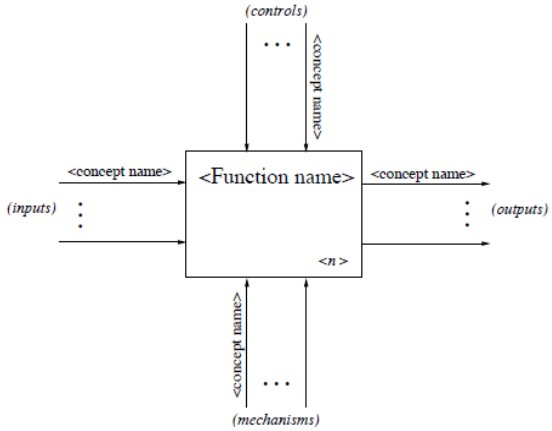


Fig. 22. The basic IDEF0 constructs (Menzel, Mayer, 1998)

IDEF0 describes any process as a series of linked activities, each with inputs and outputs. External or internal factors control each activity, and each activity requires one or more mechanisms or resources (Fig. 22).

Inputs are data or objects that are consumed or transformed by an activity. Computer or processed outputs are data or objects that are the direct result of an activity. Controls are data or objects that specify conditions that must exist for an activity to produce correct outputs. Finally, mechanisms (or resources) support the successful completion of an activity, but are not changed in any way by the activity.

The essence of IDEF0 is its hierarchical approach, in which a basic, single-activity description of the process is decomposed systematically into its constituent activities (El-Sharef, El-Kilany, 2011).

This modeling method is used for the extension of the cpm.4.CSE model described in the next subsection.

3.1.2 Process of cpm.4.CSE Extension

As shown in Chapter 2 of the thesis, based on an overview of frameworks of basic components and processes for Informatics education it was decided to choose one of them as a background for further research. We selected a process-based development of competence models to CS (Informatics) education (cpm.4.CSE) that was suggested by Zendler et al. (2016); more in Section 2.2.

Since a process model allows the development of competence models in Informatics education related to curricular requirements we selected it. This model is composed of eight subprocesses, based on the formulated objectives of the thesis; we focus only on two essential subprocesses: A2 (determines competence areas) and A3 (identifies Computer Science concepts).

The IDEF0 modeling language was used for a process model (cpm.4.CSE) (Fig. 23). The input to the subprocess A2 were Computer Science researches at the university level (e.g., Das, 2007; Tucker, 2004) and Computer Science education (e.g., ACM, 2003; 2008; ACM/IEEE-CS Joint Curriculum Task Force, 2001; Hubwieser, 2007; Fincher, 2004).

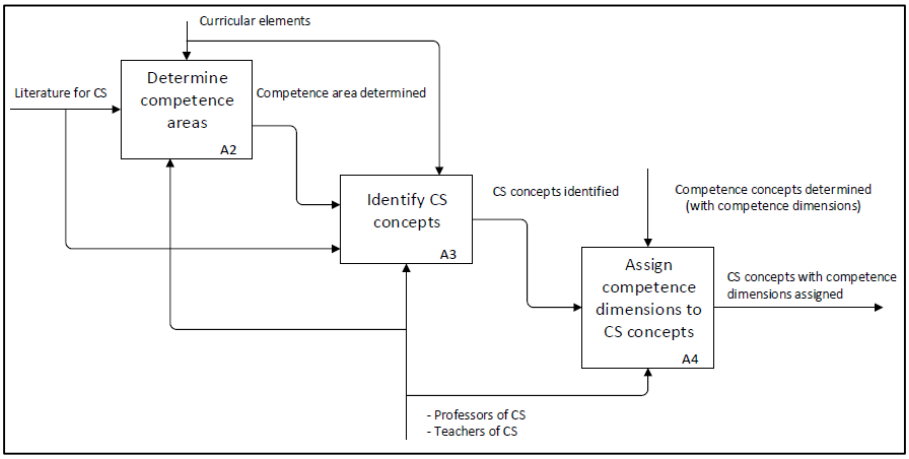


Fig. 23. The subprocesses from A2 to A4 in model cpm.4.CSE (Zendler et al., 2016)

The output of this subprocess forms four competence areas (Zendler et al., 2014):

1. **Information technology**: this competence comprises the two content concepts data and information, which are merged early due to their similar values in relation to the degree of process-related coverage and the educational accessibility.
2. **Modeling**: this competence area comprises the four content concepts problem, model, structure, and algorithm. The competence area has a high degree of process-related coverage but is not very easily accessible educationally. This is true, in particular, of algorithm. However, what is striking for this competence area is the early fusion of problem and model, whereas structure and algorithm cannot be assigned unless at some distance. This implies certain heterogeneity of the concepts on the background of their degree of process-related coverage and their educational accessibility.
3. **Computer communication**: this competence area consists of the two content concepts: computer and communication. Typical of this competence area is a low degree of process-related coverage and easy educational accessibility.
4. **Software engineering**: this competence area comprises the following seven content concepts: process, language, computation, system, test, program, and software. It characterizes content concepts whose degree of process-related coverage and educational accessibility are in the mid-range.

Subprocess A3 focuses on identifying Informatics concepts (process- and content-oriented). When designing curricula, it is necessary to know the basic content concepts as well as processes that are relevant to Computer Science (Zendler et al., 2011). The output of subprocess A3 is identified CS concepts, structured by competence areas.

For example, the competence area "Modeling" consists of CS concepts (Zendler et al., 2016):

1. Model concept. A model can be interpreted as a system (isomorphic) mapping elements of a domain to elements of a range with statements for purpose and usage;
2. Classification of models. The classification of models can be made from different points of view: area of consciousness, mode of representation, application range, and usage;
3. Diagram types. The main diagram types are class, component, activity, use case, communication, interaction, and sequence diagram;
4. Process of modeling. The process of modeling allows using diagram types to specify requirements for a software system under static, functional, and dynamic points of view;
5. Modeling languages.

The conditions of control for subprocess A3 are the same as for subprocess A1 and A2: curricular structural elements such as future life situations, necessary qualifications, or fundamental principles. Mechanisms for A3 are teachers and professors of Computer Science who are responsible for the selection of Computer Science concepts.

After long discussions with Informatics education experts and teachers it was decided (by the author of the thesis together with the supervisor) to modify the subprocesses of identification Informatics concepts for the following reasons:

1. Process model (cpm.4.CSE) is dedicated to higher education because the input to subprocess A2 is based on literature and curricular elements from colleges and universities;
2. We are interested in Informatics concepts identification for primary and secondary education, also higher education (K-12), so it is not enough to determine competencies areas. There is also the need to provide competencies and Informatics concepts/keywords. It is aimed at teachers to help them easily find and choose a particular concept-driven task. It is important to remember that Informatics is the only subject in Lithuania that teachers of primary schools have to

teach, but most likely they have never studied it. Competencies are usually defined as context-specific cognitive dispositions that are acquired and needed to successfully cope with certain situations or tasks in specific domains (Koeppen et al., 2008). Competence-oriented approaches focus on the output as a result of task-solving (performance) by learners, on several knowledge components (knowledge, skills, dispositions, attitudes), on the acquisition of competences, and on standardized methods of competence assessment (Zendler, Klaudt, Seitz, 2014). For Informatics education the competence areas are characterized by both the content concepts, and the process concepts. Informatics concepts can be considered as key components of the content of Informatics education.

A sequence of subprocesses related to Informatics concepts identification (Fig. 24) should start with the determination of competencies area - A1, identification of competencies - A2, and finish with the identification of Informatics concepts (keywords) - A3.

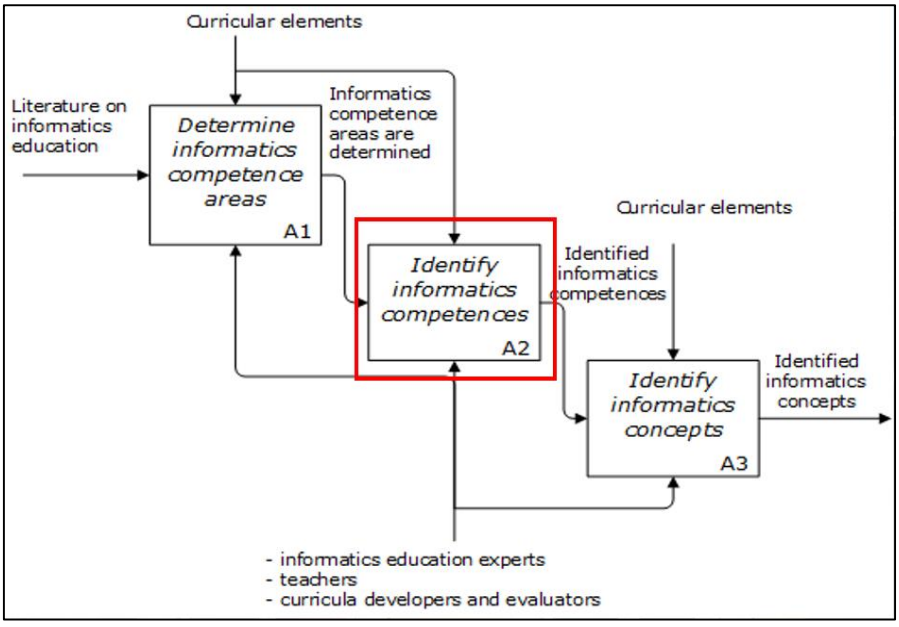


Fig. 24. Extended subprocesses for Informatics concepts identification

This extension (marked with an additional rectangle) is based on long-term practical experience of the author of the thesis together with the supervisor while using ICDT at school as the tool to introduce Informatics science. Also participation in a projects: “Network on Innovative Computing

Education" (2015-2017); "Teaching Informatics: development of activities-based model" (2016-2017); "Card-games for students to learn Informatics" (2017-2018); "Informatics in primary education" (2017-2022).

The fulfillment of modified process model is based on subprocesses called A2 (determines competencies areas) and A3 (identifies Informatics concepts) from process model cpm.4.CSE, which are documented by IDEF0 modeling language. We also focus only on the content oriented Informatics concepts for primary education. As mentioned before, we have modified a sequence of subprocesses in the process model that was suggested by Zendler et al. (2016)

The input to subprocess A1 (determines Informatics competencies areas) are literature for Informatics education at school, e.g., Australian Curriculum: Digital Technologies, v8.3, 2016; the national curriculum in England, 2013; K-12 Computer Science Framework, 2016; CSTA K–12 Computer Science Standards, 2011, and published papers, e.g., Bell et al., 2014; Caspersen, Nowack, 2013; Sysło, Kwiatkowska, 2015; Barendsen, Steenvoorden, 2016; Barendsen et al., 2016.

The control conditions for subprocess A1 are curricular structural elements (for school), which may differ from country to country.

Teachers, Informatics education experts, professors of Informatics in collaboration with education policy makers (curricula developers and evaluators), who are responsible for selecting the competencies areas are involved in mechanisms roles.

The output of subprocess A1 is determined by the Informatics competencies areas. In our context we determined six such areas: Digital content; Algorithms and programs; Problem solving; Data and information; Virtual communication; Safety and protection (see Table 11). All of them are defined as equally important.

The input to subprocess A2 (identifies Informatics competencies) are determined Informatics competencies areas and the same literature as to subprocess A1; also the same control and mechanisms elements.

The output of subprocess A2: Informatics competencies are determined. The list of competencies is provided in Table 11 (column second).

The input to subprocess A3: (identifies Informatics concepts/keywords) Informatics competencies are determined.

The output of subprocess A3: Informatics concepts / keywords are identified. The results are provided in Table 11 as well (column third).

Table 11. Outputs from subprocesses A1, A2 and A3

Competencies areas	Competencies	Informatics concepts//keywords
Digital content	Become familiar with a variety of digital content	Image representation; Sound representation; Video representation; Color representation; Character encoding;
	Use digital content	Online documents, PDF
	Create contents by using technologies	Text; Table; Cell; Formula; Chart;
	Evaluate and improve digital content	Criteria; Presentation;
Algorithms and programs	Understand the benefits of an algorithm, program	Algorithm; Searching; Shortest path; Sorting; Optimization; Sequence; Scheduling;
	Perform the sequence of actions indicated by the commands	Command; Constraint; IF condition; Variable;
	Use commands and logical operations	Operations AND, OR, NOT; Loop; Repetition;
	Create and executes programs	Program; Programming language; Coding;
	Search for debugs, tests and upgrades	Debugging; Testing; Bug;
Problem solving	Find out the problems posed by digital technology	Memory; Pattern recognition;
	Creatively use of digital technology	Multitasking; Physical devices; Robotics; Sensors; Input / Output devices;
	Select and combines of digital technologies	Parallel processing; Deadlock;
	Self-evaluation of digital competence	
Data and information	Understand the importance of data and information	Big data; Classification; Data compression; Database; Data mining; Information; Priorities;
	A targeted search for information	Data retrieval; Information search; Binary representations; Coordinates;
	Perform a variety of actions with the data: collect, store, group, sort	Sorting; Binary tree; Graph; List; Queue; Stack; String; Tree; Pattern; Table;
	Evaluate the suitability and reliability of information	Validation; Information analysis;
Virtual communication	Understand the nature of communication in the virtual space	Internet; Social networks
	Communicate by using digital technology	Mobile phone; Computer;
	Collaborate, share experiences and resources	Social networks; Cloud computing;
	Estimate the risk of virtual communication	Netiquette; E-bullying
Safety and protection	Protect devices from virus	Virus; Security;
	Protect personal data and privacy	Authentication; Copyright; License; Open Source; Legal issues;
	Manage digital identity	Self-identity; Social engineering;
	Protect environment	

Research methods from social science were used for all these processes. One of them is methodological triangulation in qualitative research that combines content analysis and the unstructured interview method.

Triangulation is a powerful technique that facilitates validation of data through cross verification from two or more sources (Carvalho, White, 1997). First, the content analysis that is defined as the systematic reading of the body of texts, images, and symbolic matter, not necessarily from the author's or user's perspective (Krippendorff, 2004) was performed. In other words, content analysis is distinguished from other kinds of social science research in that it does not require the collection of data from people. Like documentary research, content analysis is the study of already recorded information, i.e. information which has been recorded in texts, media, or physical items.

In this research work, the content analysis was conducted by analyzing documents defined as input to subprocess A1.

The second method that comprises methodological triangulation is the unstructured interview method. It is a qualitative research method in which the questions are prepared during the interview (Wethington, McDarby, 2015). In exploratory research, the unstructured interview is used as the basic tool for collecting information.

The processes of cpm.4.CSE model extension presented above and identified Informatics concepts were discussed within peer-research groups that are renown in the field; in particular, the discussion with Prof. Juraj Hromkovič (ETH Zurich University) and his group colleagues during workshops and meetings.

The concepts identification process was finally discussed during the workshop "Model of Informatics education activities", Druskininkai, Lithuania, 03-09-2016, (16 participants with experience in Informatics education).

3.1.3 Concept Map of Informatics Concepts for Primary School

As was mentioned before, teaching of Informatics at school cannot be performed without first understanding of its fundamentals.

In this section, slightly different approach to Informatics concepts will be described. This approach is based on the body of knowledge of Informatics science.

Peter J. Denning defined Informatics as "the body of knowledge dealing with the design, analysis, implementation, efficiency, and application of processes that transform information" (Denning, 1985). Later, Michael Loui defines engineering approach to Informatics as "the theory, design, and

analysis of algorithms for processing information, and the implementations of these algorithms in hardware and in software" (Loui, 1987).

Informatics is interdisciplinary at heart, because it is focused on the search for solution for problems in all areas of sciences, wherever the use of computers is imaginable. While doing so, it employs a wide spectrum of methods, ranging from precise formal mathematical methods to experienced-based "know-how" of engineering (Hromkovic, 2006).

The concept mapping method described in Section 2.1 was used to represent the relationships between Informatics concepts at primary school (

Fig. 25). Also, it is ontological point of view when presented high-level knowledge and data representation structure. Ontologies can be used to represent the structure of a domain by means of defining concepts and properties that relate them (Lhotska et al., 2013).

The first level categories of Informatics concepts are:

1. Algorithms and Programming;
2. Data, Data structures and representation;
3. Technology.

The previously identified Informatics concepts (Table 11) form the second and third levels of the concept map produced.

First level category "Algorithms and Programming" consists of three second level categories: Algorithms and Computing problems; Programming; Logic.

Category "Data, Data structures and representation" also consists of three second level categories: Data and Information; Data structuring; Data representation.

Category "Technology" consists of four second level categories: Networking; Computer architecture; Interaction; Security and privacy.

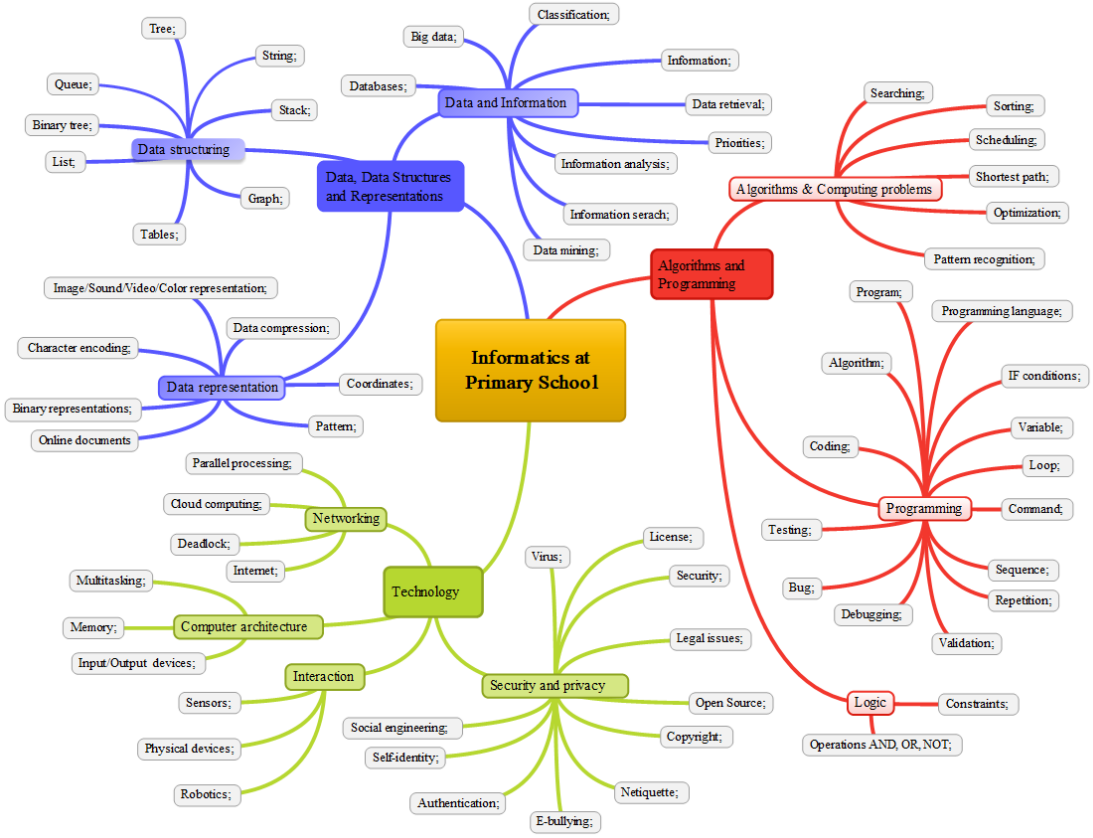


Fig. 25. Concept map of Informatics concepts for primary school

3.2 Adaptation of Two-Dimensional Categorization

Based on the analysis of the proposed two-dimensional categorization system provided in Section 2.7, we decided to adapt this system for ICDT categorization.

A logical data model is described (Fig. 26). It organizes elements of data and describes how they relate to each other and to the properties of the real world entities.

Data modeling in software engineering is the process of creating a data model for an information system by applying certain formal techniques, e.g. UML notation (class diagram). This is a static (or structural) view of the designed system, which emphasizes the static structure of the system using objects, attributes, operations and relationships.

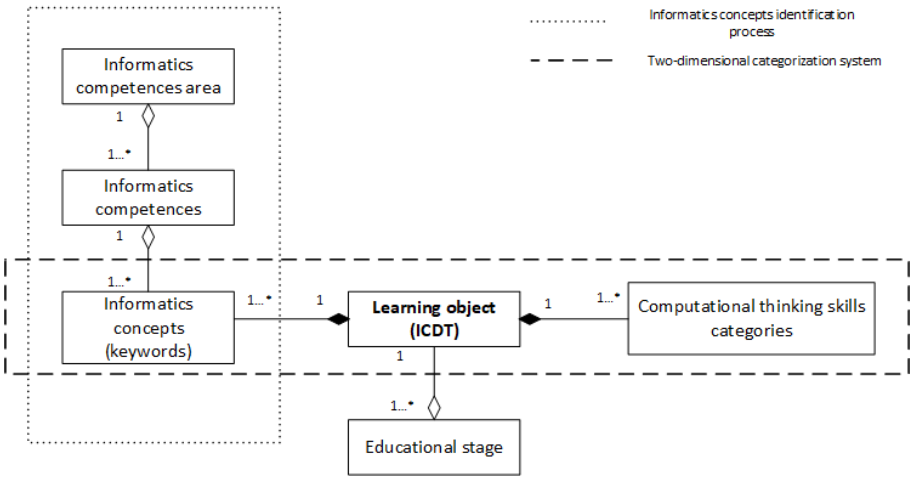


Fig. 26. Data model of two-dimensional categorization system

Two types of relations are presented in this model: Aggregation (◇——) and Composition (◆——). Aggregation implies a relationship where the child can exist independently of the parent. Example: School class (parent) and its Students (child). Delete the School class and the Students still exist. Composition implies a relationship where the child cannot exist independently of the parent. Example: House (parent) and Room (child). Rooms do not exist separately from House (Fowler, 2004).

This two-dimensional categorization system for ICDT is dynamic and can be applied to other educational levels. It depends on the results of the identification process of Informatics competencies and concepts and also on categories of computational thinking skills.

The schema with detailed information and a fragment from the example (explanation of learning task) in order to clarify the two-dimensional categorization system (Fig. 27) was prepared.

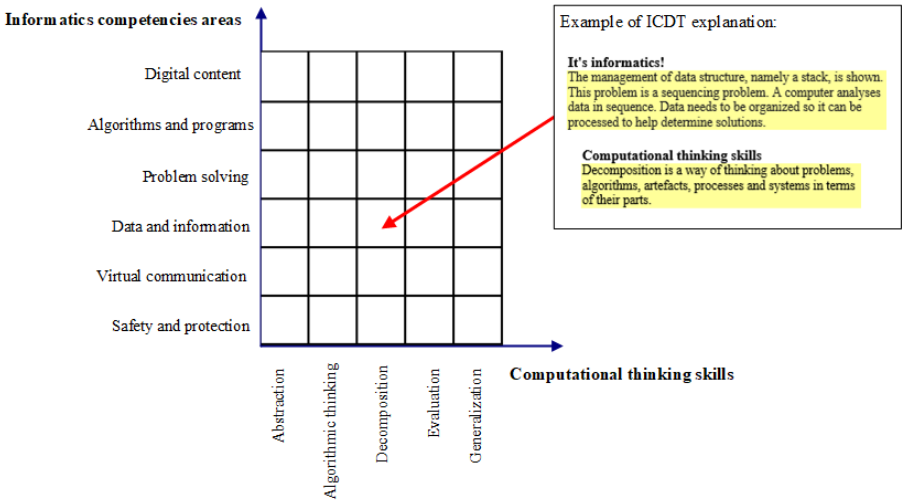


Fig. 27. Adapted two-dimensional categorization system

The learning task illustrated in example is presented in Section 2.6. The sequence of processes of ICDT creation, categorization and using for both formal and non-formal education are presented in Fig. 28.

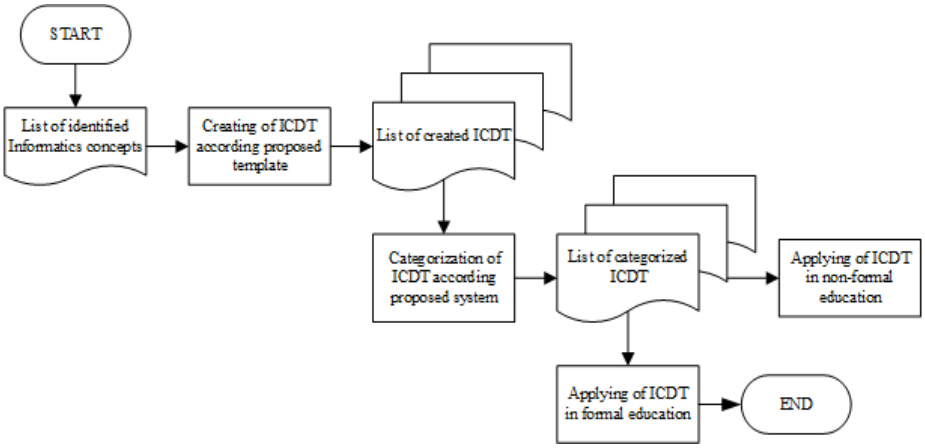


Fig. 28. Tasks creating, categorizing and using process

3.3 Modification of ICDT Template

A learning object is defined as any digital resource that can be reused to support learning (Wiley, 2000). The metadata that learning objects contain allow them to be located and retrieved, and the idea is that they can be reused in different educational contexts and can help with the specific needs of users and platforms (Morgado et al., 2018). From this point of view ICDT is a learning object and has metadata.

The IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata (LOM) (IEEE Standard…, 2002) is an internationally recognized open standard for the description of "learning objects". The LOM comprises the hierarchy of elements. At the first level, there are nine categories, each of which contains sub-elements; these sub-elements may be simple elements that hold data, or may themselves be aggregate elements, which contain further sub-elements. The semantics of an element are determined by its context: they are affected by the parent or container element in the hierarchy and by other elements in the same container. Data elements describe a learning object and are grouped into categories.

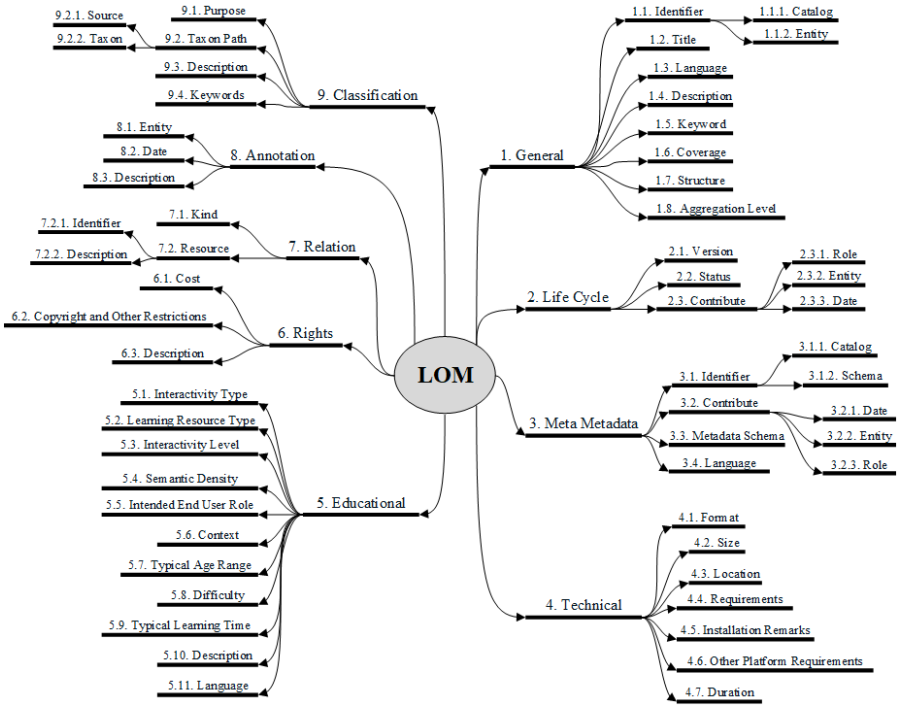


Fig. 29. The element hierarchy of the IEEE LOM standard

The LOMv1.0 base schema (Fig. 29) consists of nine such categories (IEEE Standard…, 2002):

1. The **general** category groups the general information that describes the learning object as a whole.
2. The **lifecycle** category groups the features related to the history and current state of this learning object and those who have affected this learning object during its evolution.
3. The **meta-metadata** category groups information about the metadata instance itself (rather than the learning object that the metadata instance describes).
4. The **technical** category groups the technical requirements and technical characteristics of the learning object.
5. The **educational** category groups the educational and pedagogic characteristics of the learning object.
6. The **rights** category groups the intellectual property rights and conditions of use for the learning object.
7. The **relation** category groups feature that define the relationship between the learning object and other related learning objects.
8. The **annotation** category provides comments on the educational use of the learning object and provides information on when and by whom the comments were created.
9. The **classification** category describes this learning object in relation to a particular classification system.

All data elements in LOMv1.0 base schema are optional. This means that a conforming LOM instance may include values for any data element defined in LOMv1.0 base schema.

The current version of learning task template proposed by the community of the international Bebras contest as a learning object has metadata presented in Fig. 30. It consists of four categories and 14 sub-elements. Some of them were detailed in the previous subsection. The elements of LOM used are denoted with grey boxes.

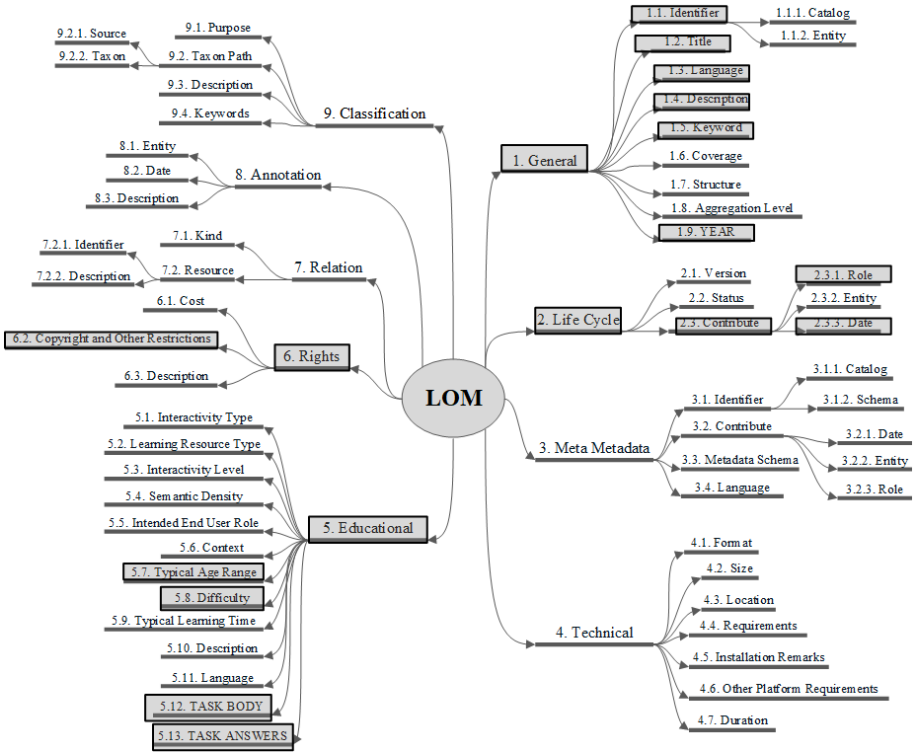


Fig. 30. Informatics learning task (international version) metadata

For the Lithuanian version we suggest to use a slightly modified structure of ICDT compared with the international version of a task (Fig. 31). This is closely related with the aim to implement the two-dimensional categorization system for learning tasks. The suggested categorization system incorporates both computational thinking skills and Informatics concepts in the classification of tasks (provided in Sect 2.7). This grey box denotes metadata that are mandatory additional attributes for ICDT. It consists of five categories and 19 sub-elements.

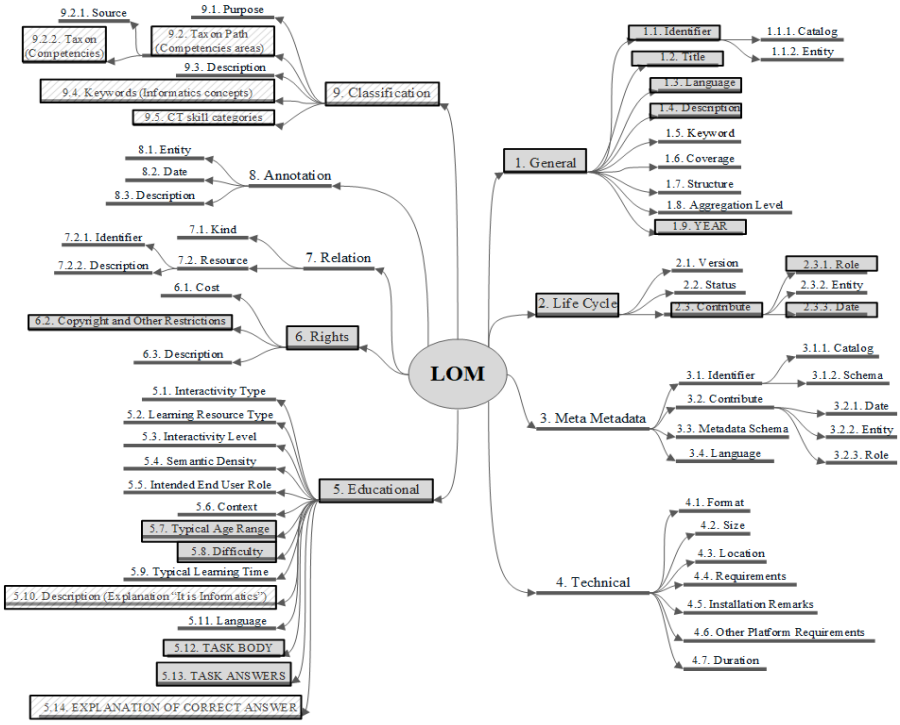


Fig. 31. Informatics concept-driven task metadata (Lithuanian version)

For the reasons of clearness for practical usage the metadata of ICDT is extracted from the whole map and is presented in more detail in Fig. 32.

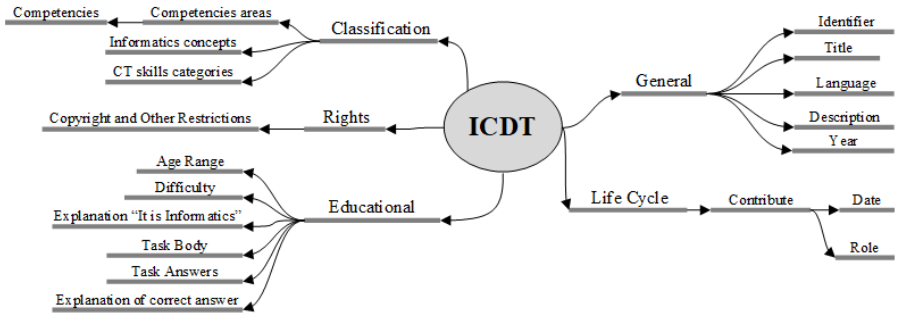


Fig. 32. Extracted metadata of ICDT

3.4 Development of CDIEM

Based on the conclusions of Chapter 2, it is reasonable to develop concept-driven Informatics education model (CDIEM) for extension of educational platform. This model is based on the following components:

1. extension of cmp.4.CSE model and results from concepts identification process (Sect 3.1.2);
2. design of ICDT template related to two-dimensional categorization system (Sect 2.7);
3. Modification of ICDT template (Sect 3.3);
4. Possibility of structural selection of ICDT in CMS.

The resulting model is defined as the flowchart presented in Fig. 33.

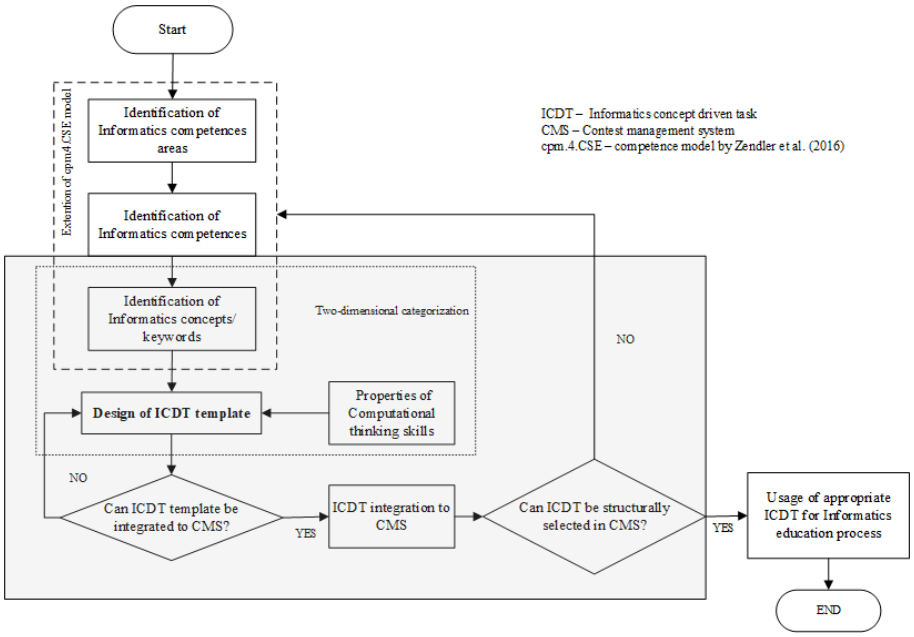


Fig. 33. Concept-driven Informatics education model

For a better understanding of the relationships between the entities of the CDIE domain, the entity relationship diagram (ERD) is developed (Fig. 34). In this case, ERD is depicted in the conceptual data model, which lacks specific details but provides an overview of the domain and how data sets relate to one another.

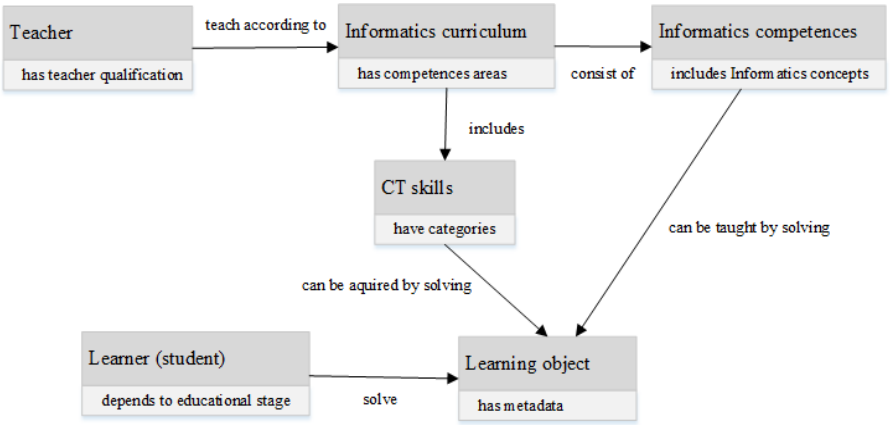


Fig. 34. Conceptual model of teaching domain

As discovered in the analytical part of the research there are no such educational platforms, which would allow teachers the structural selection of Informatics concept-driven tasks for the educational process.

3.5 Structure of the CMS Developed in Lithuania

The Lithuanian Bebras contest management system was realized in 2010 and named as the Bebras contest tool (in Lithuanian - Bebro varžybų laukas, lt.bebras.lt). The main functional requirements include the management of an information about participants (students) as well as teachers (coordinators) gather the data for solutions of tasks, organize contests and provide a detailed statistics and reports. The system was tested (more than 44 000 students entered the system in 2018) and efficiency-designed for managing contest. More than 5 500 new accounts were created for primary and secondary school students in 2017. The number of new user accounts is growing by similar additional accounts each year.

The Lithuanian Bebras CMS is based on three-tier architecture (Fig. 35) (Dagienė et al., 2017c) and it is a framework composed of MySQL relational database management system (DBMS), Apache HTTP server and PHP programming language and Linux OS. CMS use the Model-View-Controller structural pattern, which means that an application should be divided from its presentation into three main parts. In Model-View-Controller, the View component displays information to the user and together with the Controller comprises the applications user interface (Leff, Rayfield, 2001). CMS is built to be compatible with all operating systems and the latest versions of

browsers, although the use of Microsoft Internet Explorer or Microsoft Edge browsers is not recommended.

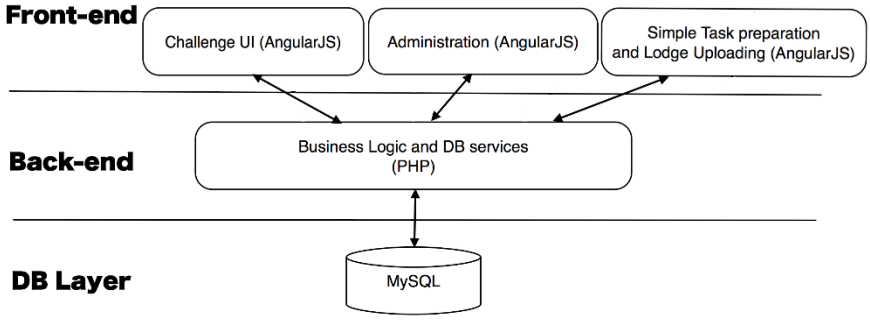


Fig. 35. The architecture of the Lithuanian Bebras CMS

The system works as a web application and consists of a set of subsystems, which has a well-designed interface:

1) System administration (security, back up, resource monitoring, etc.);
2) User management (registration system, authentication, user profile management);
3) Contest management (creation, administration and monitoring of the contest);
4) School administration (official list of all schools in Lithuania). The school list is updated in accordance with cooperation with the Centre of Information technologies in Education every year;
5) Tasks management (create, import tasks);
6) Results and communication management (participants and teachers can discuss particular tasks, preview statistical data).

Lithuanian Bebras CMS functionality is shown by using the Use case diagram (Fig. 36) (Dagienė et al., 2017b). System administrators have full access, including the management of task: creation and importation from the Bebras Lodge[2] tool (it is an authoring tool developed for coding and implementing dynamic tasks).

Teachers are provided with contest access to their schools' students and have access to the results of their students. First, the teachers register their students and then the system administrators enroll them in the system (it helps to avoid cheating). The registered teachers can confirm students'

[2] http://bebras.licejus.lt/

participation in the contest during the school-wide contest in November. Furthermore, the teachers have the opportunity to preview the tasks, participate in the discussion, and print certificates for their own students. Students have access to the contest during the Bebras week and can preview the tasks, and comment on or discuss the particular tasks after the contest week. They can see their results only after completing the contest.

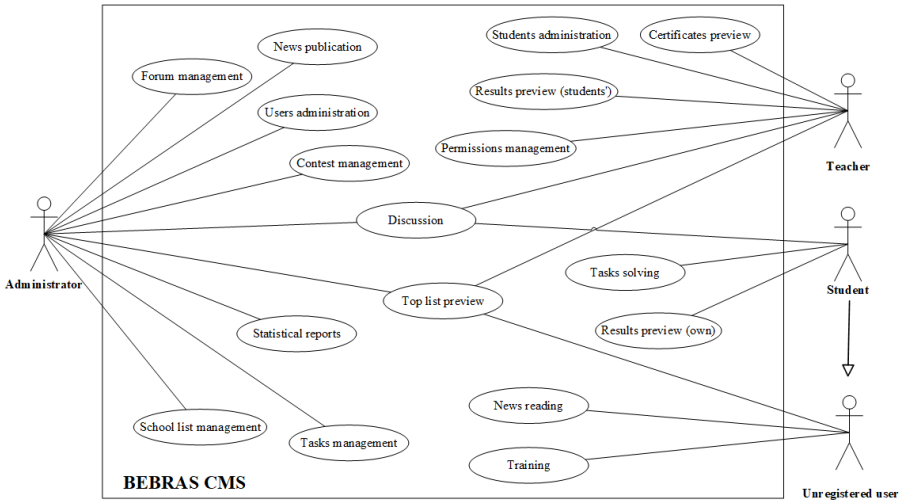


Fig. 36. Use case diagram of the Lithuanian Bebras CMS

According to its functionality (Fig. 37), CMS design is modular (consists of 11 modules). All modules are described in detail below (Dagienė et al., 2017a).

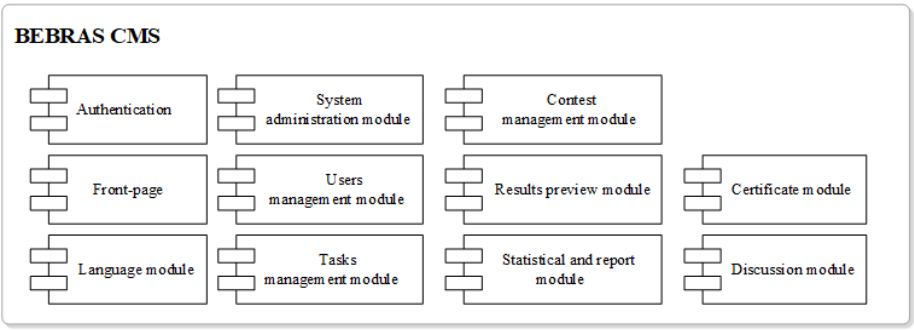


Fig. 37. The modular structure of the Lithuanian Bebras CMS

Users management module includes actions such as create, edit, confirm, delete user, preview or edit data, preview the list of participants,

archive participants, give a new password, move student to a higher class, search user, send news and reminders for the participant, give permission for the teacher to coordinate the contest, and generate the reports for students' solutions.

Contest management module involves these actions: to create and edit the task collection, set the contest date and time, preview task in the task collection, task testing, manage permission to solve task collection, participation in the contest, and answer survey about particular tasks.

System administration module consists of previewing the list of schools, editing participants and teachers' data, searching (by school title, teacher name/surname, student name/surname), and the creation of a survey for participants.

Tasks management module is designed to create, edit, copy, upload, delete, preview, export task, filter task by tags, create tags for the tasks, and search tasks by name or ID. The system supports multiple-choice questions with the opportunity to select one correct answer from four. Other questions, such as animated tasks, text input field tasks, and drag and drop tasks are imported from the Bebras Lodge tool. For multiple-choice tasks, the administrator fills in these fields: task ID, title, task description and possible answers, answer comment if it is needed, chosen task difficulty, age group, and language.

Results preview module provides the opportunity to revise the desired student and class solutions. Tasks are shown with the marked correct/incorrect answer, the student's choice, and the points he or she receives. The contest results for registered participants are stored and teachers can view them.

Statistical and report module is used to review data about the time taken to solve the tasks, answers, and the number of participants. We collect the following data about participants: name, surname, gender, grade, and school. Also, we gather data about the type of devices and browsers the participants use in the contest; how much time spent when solving the task for the first time; how much time in seconds the participant spends on every task (the sum of seconds is counted if the participant returns to solve the task). Each solution is separated by the student ID number generated by the system; therefore, all statistics can be compared. In addition, data about the students' numbers in Lithuanian schools are saved in the system as well, therefore we can compare how many students of a particular school participated.

Front-page. The module includes login to the system, a preview of participants' results according to the municipality, age group, and a report about participants' numbers in Lithuania. Training is available on the front-page without any special registration. The time of training is limited to sixty minutes and does not depend on the students' age. There is no limit to the number of answer submissions. Training results for unregistered users are displayed immediately after finishing the session.

Authentication. The module implements the user registration, login and logout function, and password reminder. Participants are able to login with a Facebook account. When a user registers with the system for the first time, the system automatically sends an email with the link, and the user has to approve it. An email is a required field on the systems registration form for purposes of authentication and personal data. Primary school students are faced with the problem of not having a personal email account. For this reason, their teachers can upload the list of participants to the system. The system generates passwords and usernames automatically from the uploaded file.

Certificate module. The system automatically generates the filled certificate that is prepared by an administrator after the contest is completed. Teachers get certificates for organizing and coordinating the contest. Students who participated in the first or second round of the contest receive certificates for participation. About 10% of the best students from each age group get winners' diplomas. These diplomas are printed and students receive them during the awards ceremony.

Language module enables localization and adapts the system for a specific language by translating resources.

Employing the **discussion module** users can follow discussions (read, comment, delete, edit).

In the case of the Lithuanian Bebras CMS (lt.bebras.lt), a relational database (of tasks management subsystem) is created to store details of tasks (Fig. 38). The data are stored in different tables and relations are established using the primary keys; below the information engineering notation is used to represent a logical data model developed.

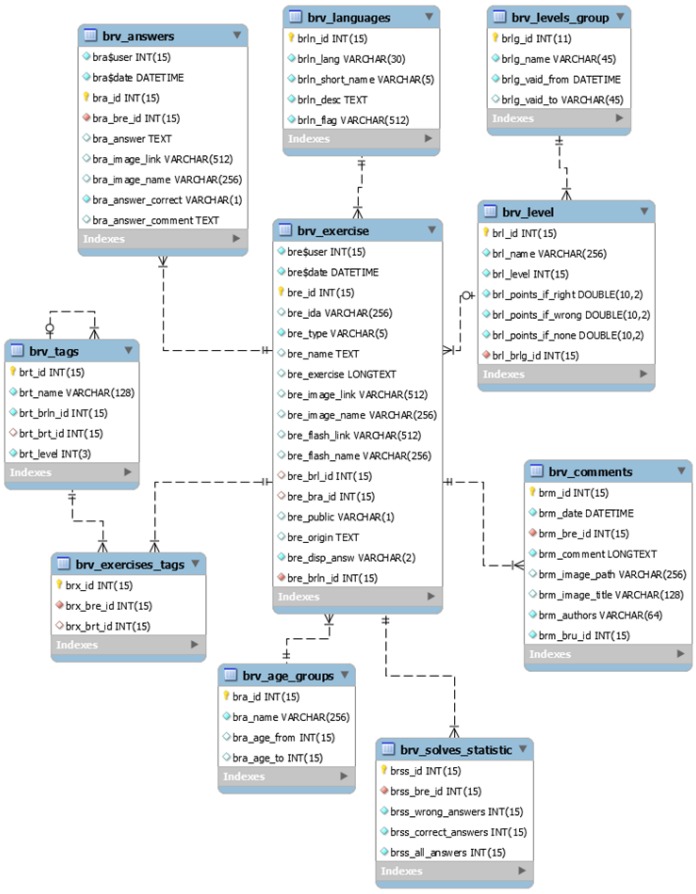


Fig. 38. The relational database structure underlying the task management subsystem

The Lithuanian Bebras CMS is mainly used as a tool for contest management. Sometimes it also used for training teachers or systems developers to provide them with practice, understanding the contest policy. Unfortunately, the Lithuanian system does not provide teachers with the possibility of preparing their own contest to use in the educational process.

The Bebras CMS also is used as a repository. At the end of 2018, more than 2 300 tasks were collected in the platform. At the same time, this platform lacks the possibility to structurally select appropriate ICDT and to form ICDT's collections in order to use them for further educational process (for formal, informal and non-formal education).

Regarding to developed CDIEM, there is a need to extend the educational platform. For this reason, in the next subsection, the process of the educational platform extension design is described.

3.6 Design of Educational Platform Extension

The design of educational platform extension presented here is aimed at the implementation of concept-driven Informatics education model. This extension is based on:

1. Integration of additional attributes of the ICDT template into the Lithuanian Bebras CMS;
2. Creation of a new module for the structured selection of ICDT in the Lithuanian Bebras CMS.

The structure of the improved ICDT described in Section 3.3 is used to implement the first step. We provide an improved version of the relational database (Fig. 39) in order to show differences between this one and the existing relational database of the task management subsystem in the Lithuanian Bebras CMS (Fig. 38).

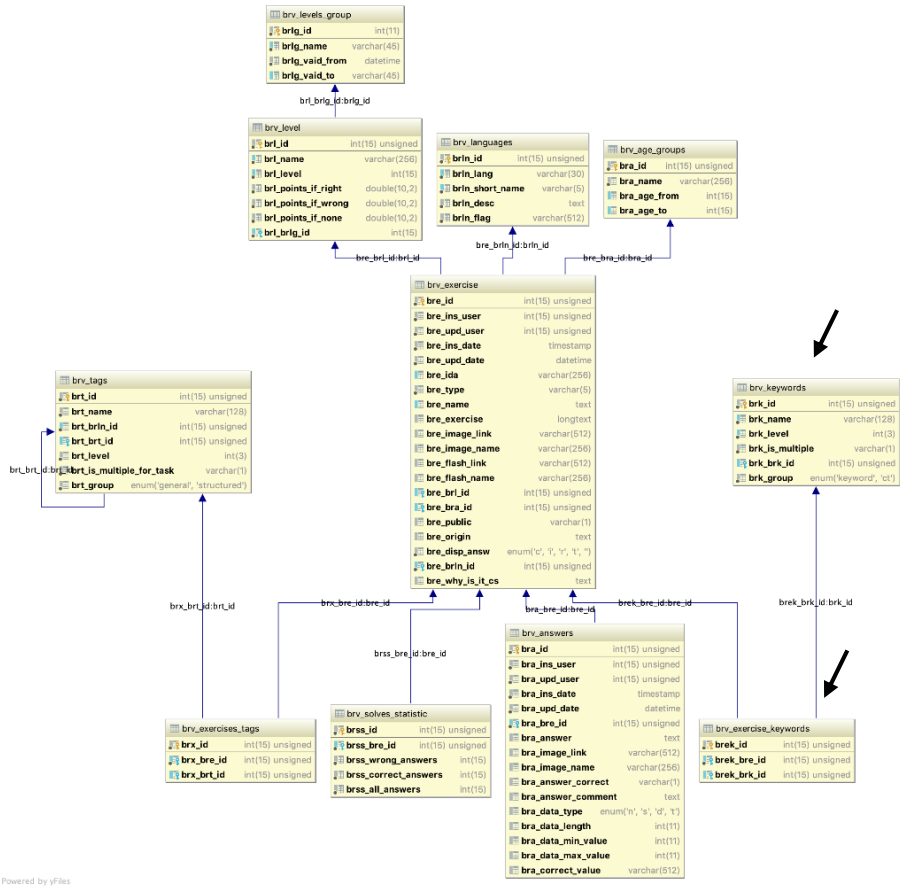


Fig. 39. The relational database model underlying the EEP

The relational database (Fig. 39) was supplemented by two tables (denoted by arrows): *brv_exercise_keywords* and *brv_keywords.* These two tables related to the implementation of the two-dimensional categorization of tasks are created. Table *brv_exercise* was also supplemented by one attribute *bre_why_is_it_cs*, which include records of explanation “It is Informatics”.

The use case diagram of main modules related to tasks management in the current version of CMS is shown in Fig. 40.

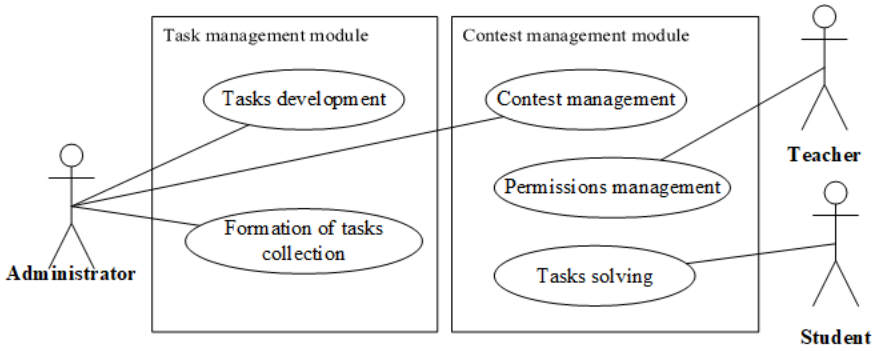


Fig. 40. Use case diagram of main modules in the current version of CMS

It shows that the teacher can give permission to a student to participate in the contest. Meanwhile the existing system does not provide functional features of task management, selection of tasks in order to form a collection of them, which are appropriate for a particular use in the educational process. All the most important features related with tasks and contest management are available only for the platform administrators.

For this purpose, there is a reasoned proposal to develop a new module (Task selection module) in the current version of the Lithuanian Bebras CMS. The Lithuanian Bebras CMS is presented in Subsection 3.5. An extended structure of CMS is in Fig. 41.

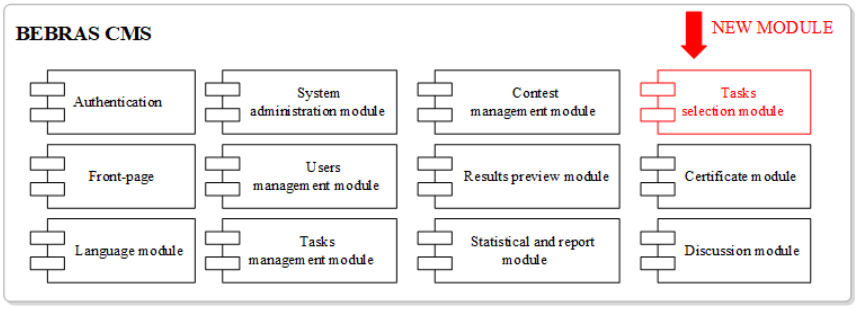


Fig. 41. New module in the extended CMS

The Use case diagram of the tasks selection module (Fig. 42) shows that there are two users (teacher and student) and their interactions with the module.

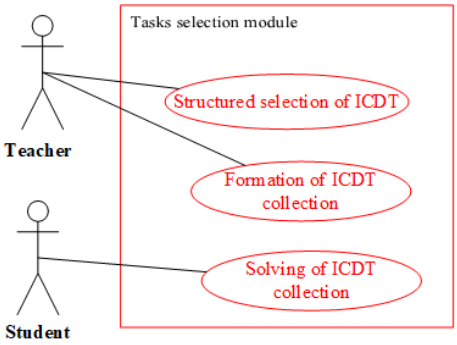


Fig. 42. Use case diagram of task selection module

The activity diagram presents a dynamic behavior of the system, i.e. the actions of the teacher with the task management module. The most important function of the task selection module is that the teacher can perform structural selection of ICDT. The structural selection of the tasks is presented in Fig. 43. The teacher can form a collection of tasks that can be additionally filtered according to CT skills.

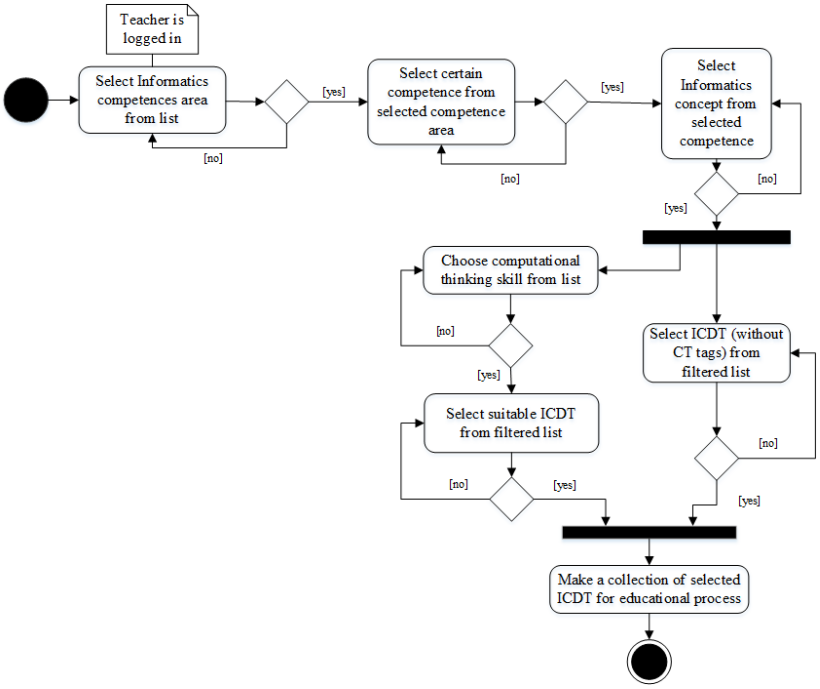


Fig. 43. Activity diagram of structural task selection process in the module

The activity diagram of the whole process for tasks selection is presented in Fig. 44.

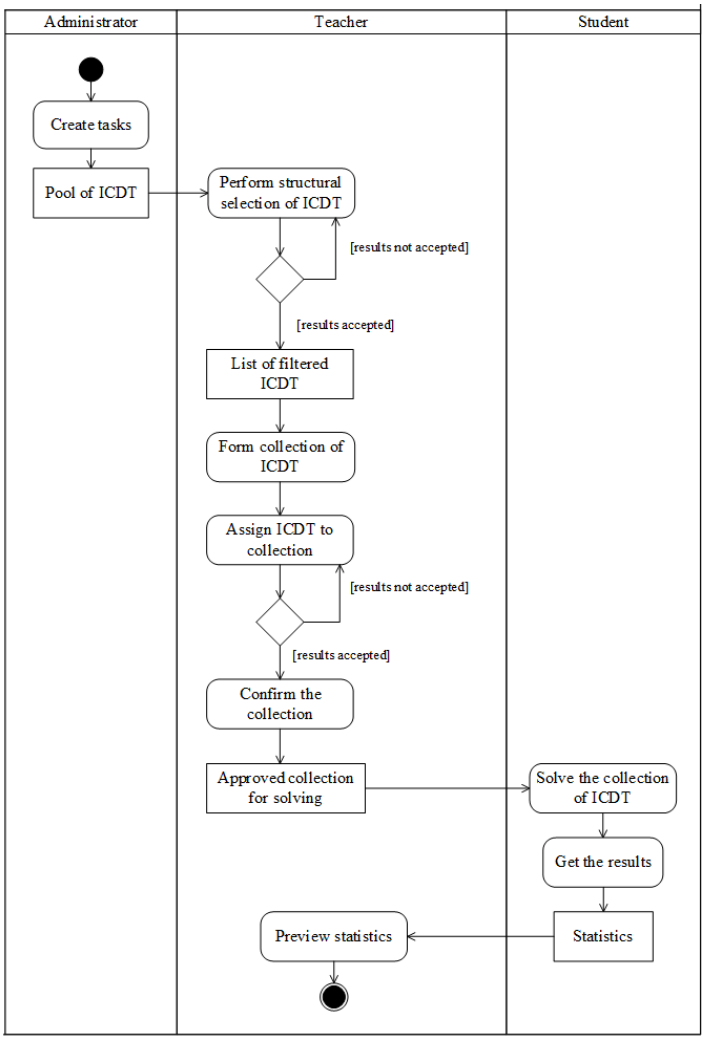


Fig. 44. Activity diagram of the whole task selection process

3.7 Implementation of the Prototype of the Designed Module

In order to verify the designed task selection model, the prototype was created. The concept-driven Informatics education model is implemented within the prototype of the EEP that was developed on the basis of the existing Lithuanian Bebras CMS.

In this extended platform the selection of tasks is realized for two competencies areas: Algorithms and programming, and Data and information. It is based on the results presented in Section 2.7.2.

First, the additional ICDT attributes were integrated into the task management module. Then the administrator should mark appropriate Computational thinking categories and Informatics competencies and concepts for each task in the task creation mode (Fig. 45). Informatics competencies areas, competencies and concepts are presented in a hierarchical structure like in the hierarchical concept map that contains the domain of knowledge on the top of the map.

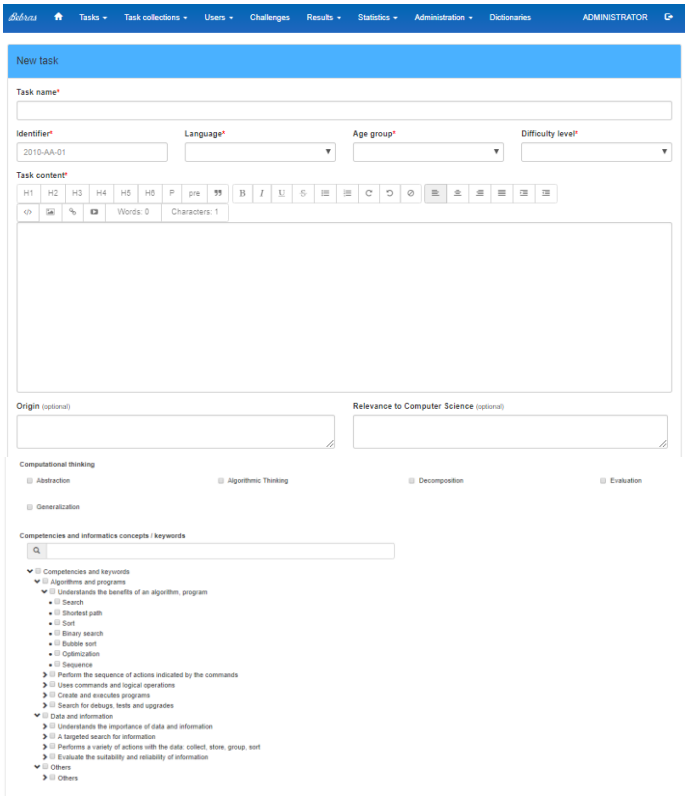


Fig. 45. Task creation mode of the prototype

After integrating additional ICDT attributes to the task management module, the teacher can fulfill a structural selection of ICDT for the educational process and decide which of CT categories and Informatics competencies should be included in a particular ICDT (Fig. 46).

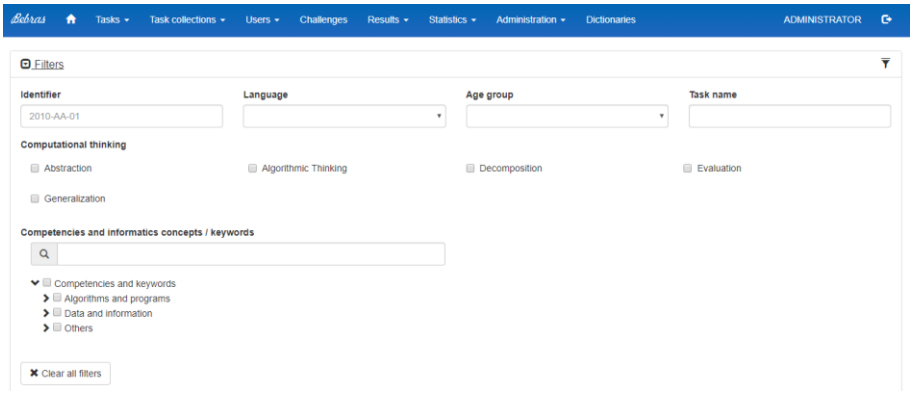


Fig. 46. Task selection mode of the prototype

In case the teacher wants to form a collection of tasks, she or he should start to create a new task collection and then fill in all required fields (Fig. 47).

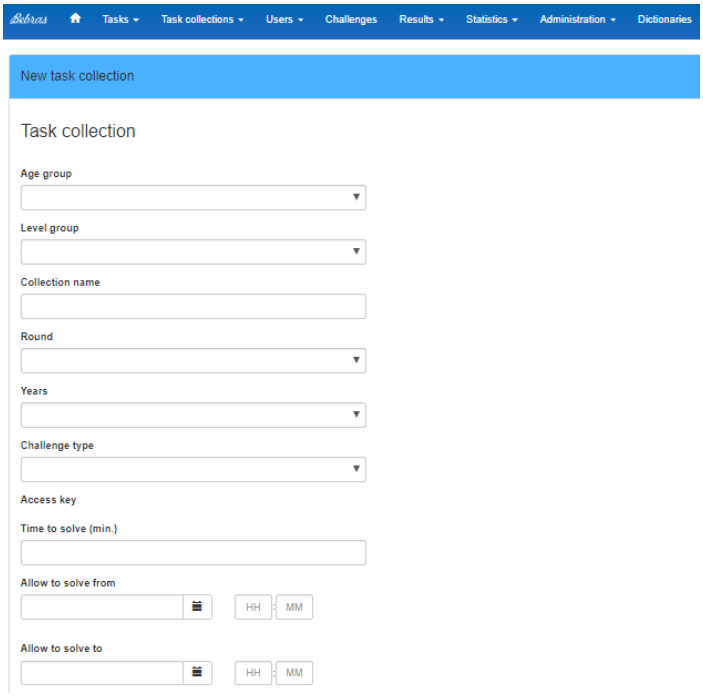


Fig. 47. Formation of tasks collection

Selected tasks can be easily added to the collection by clicking on the button + at the task collection mode (Fig. 48). Tasks can be added by selecting one of three difficulty levels (easy, medium, hard).

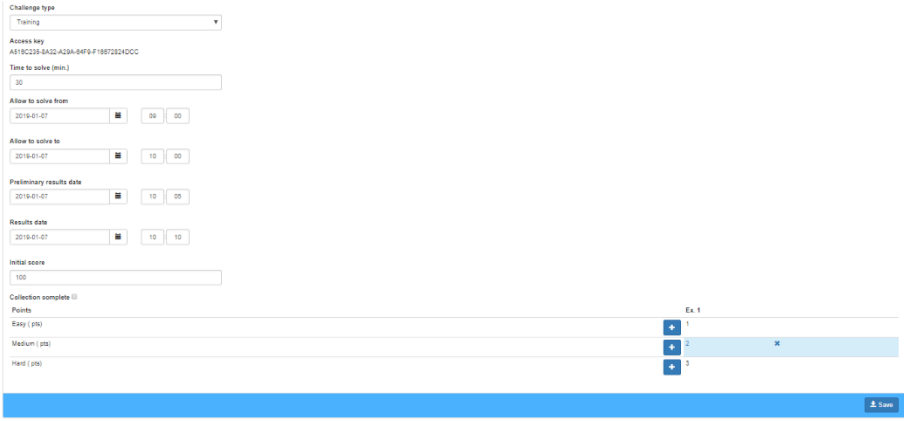


Fig. 48. Example of tasks collection formation in prototype

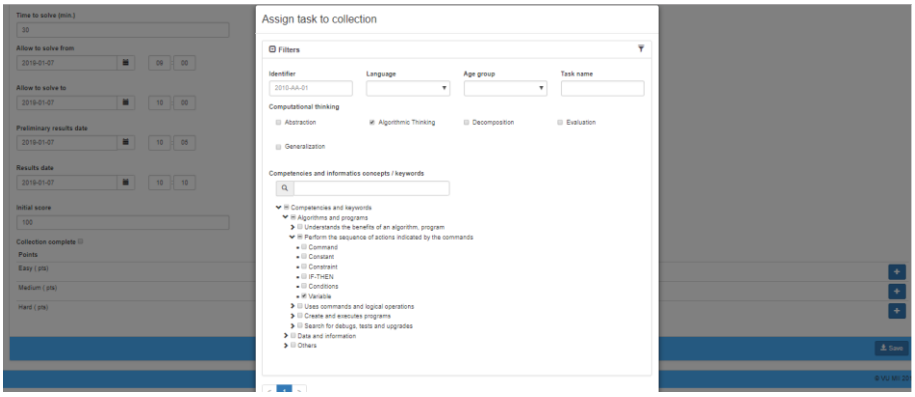


Fig. 49. Example of task assignment to collection in prototype

An example of task assignment to the collection is presented in Fig. 49. All appropriate tasks can be additionally filtered by performing CT and Informatics concepts selection.

3.8 Summary

In this section the main components of concept-driven Informatics education model for the educational platform extension were described.

First, the extension of existing cmp.4.CSE model was made. This model was selected for the following reasons: (1) clearly indicated steps of the whole educational process; (2) possible adaptation to school level; (3) provides framework of competencies and concepts.

The results of the identification process of Informatics competencies and concepts were obtained by performing methodological triangulation in qualitative research that combines content analysis and the unstructured interview method. Six competency areas were determined.

Second, the adaptation of this two-dimensional categorization system for ICDT was performed. The proposed categorization system incorporates both Informatics concepts and computational thinking skills. This categorization system is dynamic and can be applied not only at primary school but also at higher levels. It depends on the results of the identification process of Informatics competencies and concepts and also on the selected categories of computational thinking skills.

Third, the ICDT template that supports the LOM metadata structure and is related to the two-dimensional categorization system was designed.

All these components are integrated into the model of CDIE for primary school and implemented within the extension of the Lithuanian Bebras CMS (educational platform).

4. EXPERIMENTAL PART

In this part, the performed evaluation of the two-dimensional categorization system for Informatics learning tasks will be described. While the first evaluation was of qualitative nature, the developed EEP was evaluated by experts using quantitative approach based on selected quality in use criteria and fuzzy numbers.

4.1 Evaluation of Categorization System

The evaluation of two-dimensional categorization system was planned during the annual international Bebras contest community meeting in Bodrum, Turkey (May 2016), with representatives from almost every country in the community. There were around 80 members of the Bebras community present and the early version of the proposed system was explained and exemplified.

The Bebras community members were given three tasks (Beaver the Alchemist; Reaching the target; Beaver tutorials) as example to categorize according to a proposed version of the categorization system. The members were asked for each task to fill in the table like that presented in Table 12. And also answer the question: How would you categorize the following tasks utilizing the new (proposed) classification?

Table 12. Example of the questionnaire

Task name	**Beaver the Alchemist**				
Concept	Algorithms and programming	Data and data structures	Computer processes and hardware	Communi-cation and networking	Systems and society
Tick					
CT Skill	Abstraction	Algorithmic thinking	Decom-position	Evaluation	Generalization
Tick					

Also, the members were asked for additional comments on questions like: What is missing? Do categories overlap? Is it possible to select only one of them? Is this too complex?

The feedback was collected both verbally (summary of comments) and in writing (filling the questionnaire). The feedback and comments of each member were used to refine the system.

In particular, the experience of sharing the categorization illuminated some of the points raised below:

1. "Clear illustration of computational thinking skills with examples is needed as we cannot assume that any knowledge of these is shared in the community";
2. "Keywords are essential both to illustrate the Informatics content domains and the computational thinking skills to assist categorizers";
3. "Categorizers need to focus on the experience of the student solving the problem and not the task setter (expert) in assigning both the concept and computational thinking skills".

The proposed task categorization system was applied for real practice for first time in 2016 by UK Bebras organizers (University of Oxford). They have prepared a booklet with answers and explanations of learning tasks for teachers and students[3]. Later they repeated it for the 2017 and 2018 Bebras tasks.

In 2017 and 2018 this categorization system was also used and tested by the organizers of the TCS Oxford Computing Challenge[4]. It is an online challenge that asks the students invited from the UK and English-speaking international schools around the world to solve tasks using computational thinking skills and then provide coded solutions. After each task there is an example answer, an explanation of how the answer could be obtained plus a section on how the tasks are related to computational thinking (Fig. 50). They have mapped each task to up to three computational thinking skills and add also to a particular Informatics domain.

[3] http://www.bebras.uk/answer-booklets.html
[4] http://www.tcsocc.uk/prepare.html#examples

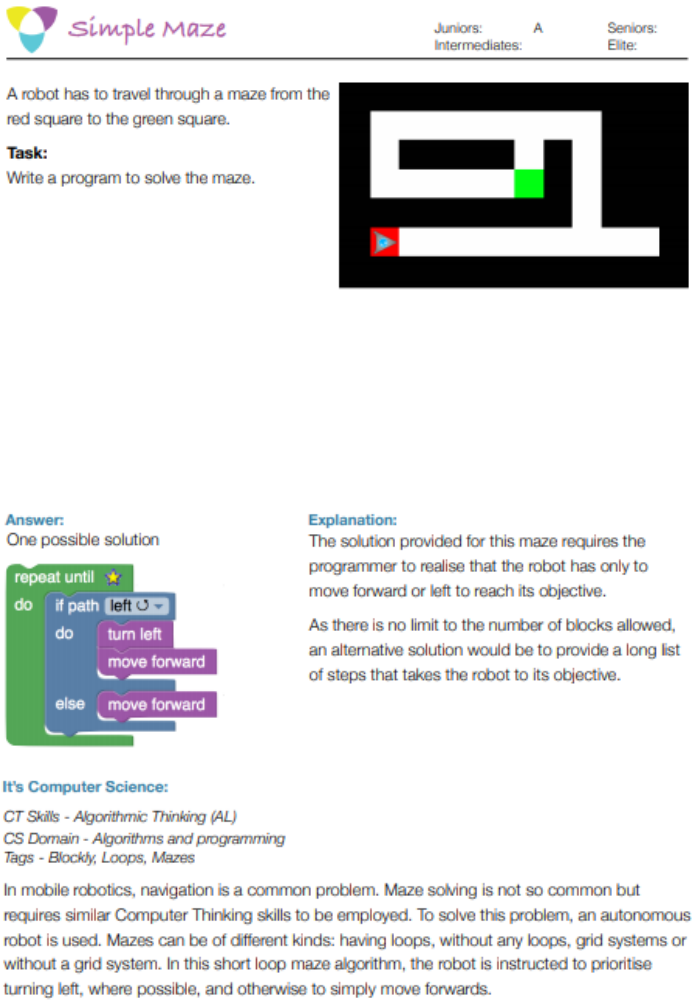


Fig. 50. Example from the 21 tasks in the TCS Oxford Computing Challenge, 2017

In 2019 Csizmadia, Standl and Waite published research based on proposed two-dimensional categorization system. Their contribution was to present a new mapping tool which can be used to review classroom activities in terms of both computational thinking and constructionist learning. For the tool, they have reused existing definitions of Computer Science concepts and computational thinking concepts (from the two-dimensional categorization system) and combined these with new constructionism matrix.

4.2 Evaluation of Extended Educational Platform

The CDIE model, proposed in the previous section, is aimed to support the concept-driven approach to Informatics education at primary school.

To assess the quality of the CDIEM implemented by the extension of the educational platform in respect of usage, the experts' evaluation method is selected.

According to Oppermann, Reiterer (1997) the expert evaluation methods draw upon expert knowledge to make judgments about the usability of the product for specific end users and tasks.

4.2.1 Quality in Use Model

In order to determine how much the proposed EEP allows an effective and structural ICDT selection (which opens opportunities to get appropriate ICDT and use them in real educational practice), an interview of experts was conducted.

At present, the EEP lacks some features and data that would allow us to carry out a full range engineering experiment. Pilot data are produced using the prototype.

A quality model provides means to control software quality (Kan, 2002). It usually defines quality attributes that good software should have and can associate metrics or a methodology to assess the level of quality. According to Gasperovic and Calpinskas (2006) from the technological point of view, one of the quality criteria of the learning software is quality in use and it is an evaluative characteristic of software obtained by making a judgment based on the criteria that determine the worthiness of software for particular users.

According to the standard, the quality in use model (ISO/IEC 25010:2011) is selected to be used in this thesis. It is composed of five characteristics (Fig. 51), which are further subdivided into sub-characteristics. The later can be measured when a product is used in a realistic context.

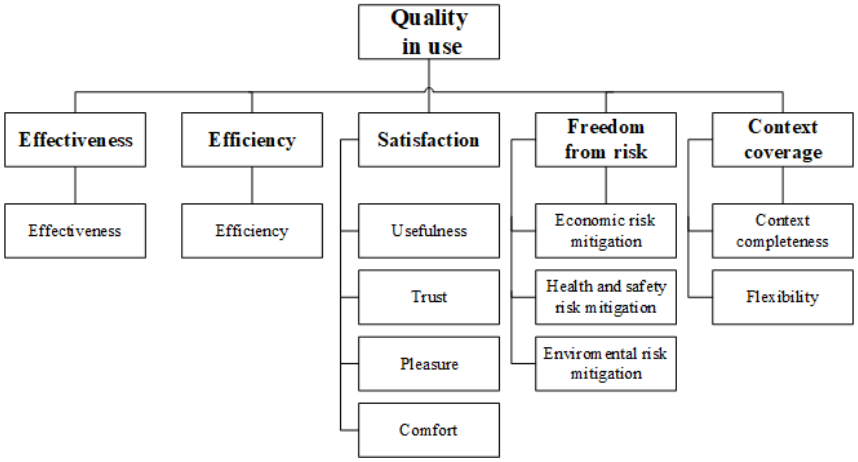


Fig. 51. The quality in use model according to ISO/IEC 25010:2011

The quality in use is the user's perspective view of the quality of a system, and is measured in terms of the result of using the system (i.e. how people behave and whether they are successful in their tasks), rather than the properties of the system itself. The output can be measured as effectiveness, productivity, and satisfaction of the users (ISO/IEC 25010:2011).

Definitions of each characteristics and subcharacteristics of the quality in use model are presented in Table 13.

Table 13. Quality in use model characteristics and subcharacteristics

Characteristics	Sub-characteristics	Definition
Effectiveness		Accuracy and completeness with which users achieve specified goals.
Efficiency		Resources expended in relation to the accuracy and completeness with which users achieve goals.
Satisfaction		Degree to which user needs are satisfied when a product or system is used in a specified context of use.
	Usefulness	Degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use.
	Trust	Degree to which a user or other stakeholder has confidence that a product or system will behave as intended.
	Pleasure	Degree to which a user obtains pleasure from fulfilling their personal needs.

	Comfort	Degree to which the user is satisfied with physical comfort.
Freedom from risk		Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.
	Economic risk mitigation	Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.
	Health and safety risk mitigation	Degree to which a product or system mitigates the potential risk to people in the intended contexts of use.
	Environmental risk mitigation	Degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use.
Context coverage		Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.
	Flexibility	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements.
	Context completeness	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.

In this work, four criteria (Table 14) were formulated to evaluate the quality in use of the EEP: (1) Effectiveness, (2) Efficiency, (3) Flexibility, (4) Context completeness. In this thesis, we do not measure two characteristics (with their subcharacteristics): Satisfaction and Freedom from risk. They are not quantifiable qualities in a case of the educational platform, which is specific in respect of the users and purpose.

Table 14. EEP evaluation criteria (adapted from ISO/IEC 25010:2011)

Characteristics	Subcharacteristics	Definition
Effectiveness		Appropriate ICDT are provided to teachers, which allows them to form a collection of ICDT that can be solved by the student. It helps the student pursue competencies provided in the Informatics curriculum.

Efficiency		Method proposed in the extended educational platform allows teachers to save time in selecting the appropriate ICDT for the educational process.
Context coverage	*Flexibility*	In the extended educational platform proposed method allows teachers to modify the process of ICDT selection.
	Context completeness	ICDT template proposed in the extended educational platform can be used in all the prescribed contexts of use.

4.2.2 Experts' evaluation

Since the expert evaluation method depends on the skill of the expert, the following competence requirements for their selection were defined by the author:

1. no less than five-year experience in the field of Informatics education;
2. no less than three-year experience of teaching Informatics;
3. at least two scientific papers published in the field of Informatics or Informatics engineering;
4. Master's degree (or Doctor's degree) in Informatics or Informatics engineering.

In order to reduce the subjectivity of evaluation due to the expert's personal assessment a detailed presentation and instructions were used to evaluate the quality of the EEP.

In many practical situations, decision makers (experts) may be reluctant or unable to assign exact numerical values to make comparison judgments. Therefore, for resolving the uncertainty and imprecision of software evaluation, the comparative judgments of a decision-maker are represented as triangular fuzzy numbers (Chang, Wu, Lin, 2008). Fuzzy sets theory oriented towards the rationalization of uncertainty. The application of uncertainty lets the experts evaluate not only one point but an appropriate range of values (Byrne, 1995). A fuzzy set is a class of objects with a graded continuum of membership and is characterized by a membership function, which assigns to each object a membership grade between zero and one. Since each number represents the subjective opinion of decision makers and is an ambiguous concept, fuzzy numbers work best to consolidate fragmented expert opinions. A triangular fuzzy numbers (Fig. 52) is denoted simply as (L, M, U), the parameters L, M and U denote the smallest possible

value, the most promising value and the largest possible value (Chang, Wu, Lin, 2008).

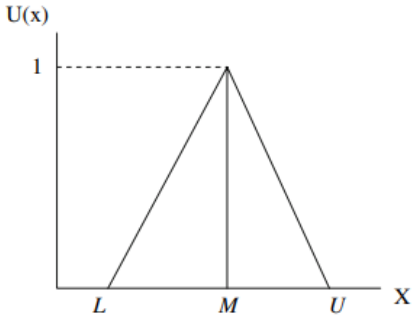


Fig. 52. Triangular fuzzy numbers

In this thesis, the experts used the linguistic variables “very small”, “small”, “average”, “good”, and “very good” to establish the ratings (values) of the quality criteria.

The triangular fuzzy numbers (Table 15) were chosen to use for this evaluation.

Table 15. Triangular Fuzzy numbers values

Linguistic term	Triangular Fuzzy numbers value
Very good (VG)	(0.8; 1.0; 1.0)
Good (G)	(0.6; 0.8; 1.0)
Average (A)	(0.4; 0.6; 0.8)
Small (S)	(0.2; 0.4; 0.6)
Very small (VS)	(0.0; 0.2; 0.2)

A questionnaire, consisting of four questions, based on the ISO/IEC 25010:2011 quality in use model was developed for experts’ evaluation. Table 16 presents the questions and the corresponding answer options.

Table 16. Questions of the developed questionnaire

1. Appropriateness of the achieved goals (Effectiveness)	
Very good (VG)	86 -100 % accurate
Good (G)	67 - 85 % accurate
Average (A)	50 - 66 % accurate
Small (S)	33 - 49 % accurate
Very small (VS)	0 - 32 % accurate

2. The time spent to achieve the goals compared with the time spent to achieve goals without the prototype (Efficiency)	
Very good (VG)	86 -100 % time saved
Good (G)	67 - 85 % time saved
Average (A)	50 - 66 % time saved
Small (S)	33 - 49 % time saved
Very small (VS)	0 - 32 % time saved
3. Flexibility of the achieved goals	
Very good (VG)	86 -100 % accurate
Good (G)	67 - 85 % accurate
Average (A)	50 - 66 % accurate
Small (S)	33 - 49 % accurate
Very small (VS)	0 - 32 % accurate
4. Context completeness of the achieved goals	
Very good (VG)	86 -100 % accurate
Good (G)	67 - 85 % accurate
Average (A)	50 - 66 % accurate
Small (S)	33 - 49 % accurate
Very small (VS)	0 - 32 % accurate
Please comment all the options (except the option "Very good")	

After defining the evaluation criteria, the interviews with the selected experts were conducted. The designed EEP prototype was presented and experts were asked to choose the given values of evaluation criteria. Experts' evaluation results are presented in Table 17.

Table 17. Experts' evaluation results

No.	Criteria	Experts						
		E1	E2	E3	E4	E5	E6	E7
1.	Effectiveness	VG	VG	VG	VG	G	VG	VG
2.	Efficiency	VG	G	G	VG	VG	VG	G
3.	Flexibility	G	VG	VG	VG	VG	VG	G
4.	Context completeness	VG	VG	VG	G	VG	VG	A

Different decision making theories (e.g. Fuzzy, AHP) are applied to obtain final evaluation measures. The influence of uncertainty could be evaluated in different ways by applying the theory of Fuzzy numbers or mathematical statistics methods. According to Kurilovas and Serikovienė, 2013, the fuzzy numbers are applicable to evaluate the quality of learning software. After applying the multiple criteria decision making technique

(Skūpienė, 2010) and using fuzzy numbers linguistic values, the experts' evaluation was turned into a numerical scale, which specifies the expression of criterion (Table 18). Other calculations use the following middle values of the triangular fuzzy numbers: VG – 1.0, G – 0.8, A – 0.6, S – 0.4, VS – 0.2.

Table 18. Experts' evaluation results converted into numerical values

No.	Criteria	Weight	Experts							Average (a_i)
			E1	E2	E3	E4	E5	E6	E7	
1.	Effectiveness	0.25	1	1	1	1	0.8	1	1	0.9714
2.	Efficiency	0.25	1	0.8	0.8	1	1	1	0.8	0.9143
3.	Flexibility	0.25	0.8	1	1	1	1	1	0.8	0.9429
4.	Context completeness	0.25	1	1	1	0.8	1	1	0.6	0.9143

All decision-makers are equally important, their estimates x_{ij}, where $i = 1, 2, ..., n$ (number of criteria) and $j = 1, 2, ..., m$, (number of experts), have the same weight $d_j = 0.1429$, and

$$\sum_{j=1}^{m} d_j = 1, \text{where j} > 2. \quad (1)$$

Arithmetic means of each criterion a_i were calculated (Table 18). Every criterion in the set of evaluation is of equal importance, so their weights s_i are equal to 0.25 and

$$\sum_{i=1}^{n} s_i = 1, \text{where } i > 2. \quad (2)$$

Finally, an overall evaluation of the quality of the EEP was made. To do that, the following function $f(x)$, is calculated:

$$f(x) = \sum_{i=1}^{n} s_i \cdot a_i \quad (3)$$

Applying the formula, the final result is:

$$f(x) = 0.25 \times 0.971429 + 0.25 \times 0.914286 + 0.25 \times 0.942857 + 0.25 \times 0.914286 = 0.935714$$

To summarize the procedure, the overall evaluation result depends on triangular fuzzy numbers conversion to linguistic variables values (Table 15), and is transformed into a linguistic variable. Then it was determined that the overall evaluation of the extended educational platform is very high, it corresponds to 93.57% of the absolute quality (i.e. 100%).

The particular results also show that the effectiveness of EEP is very good (97.14 %), efficiency – very good (91.43 %), flexibility – very good (94.29 %), context completeness – very good (91.43 %).

4.3 Summary

Two different evaluation processes were described in this chapter. The first evaluation was selected for the two-dimensional categorization system, one of components of the concept-driven Informatics education model. This evaluation was iterative, i.e. it consists of a repetition of a process in order to generate a sequence of outcomes. Each repetition of the process is a single iteration, and the outcome of each iteration is then the starting point (input) of the next iteration.

The results have showed that chosen keywords are essential both to illustrate the Informatics content domains and the computational thinking skills and to assist categorizers; also this categorization system is appropriate and can be used for similar learning tasks in other Informatics contests.

After the experts' evaluation of the quality in use of the developed extended educational platform, it has been determined that it is of very high quality in terms of the following criteria:

1. Effectiveness – while appropriate ICDT are provided to the teacher, it allows teachers to form a collection of ICDT that can be presented to the student. It helps the student to pursue competencies provided in the Informatics curriculum;
2. Efficiency – a task selection method proposed in the EEP allows the user to save time while selecting the appropriate ICDT for a particular topic;
3. Flexibility – the proposed method also allows the user to modify the process of ICDT selection in the EEP;
4. Context completeness – the ICDT template applied within the EEP can be used in all the specified contexts of use, thus ensuring context conformity.

CONCLUSIONS

1. After analyzing the impact of the Informatics concept-driven approach, and reviewing Informatics concepts, the following conclusions and results have been obtained:
   a. The Informatics concept-driven approach is helpful to the learner to gain conceptual knowledge, not only procedural knowledge.
   b. The overview of frameworks of basic components and processes for Informatics education shows that the most promising framework for further devotement of concept-driven Informatics education is the cpm.4.CSE. The main reasons are that it clearly indicates steps of the whole process and is closely related with the determination of Informatics competences and concepts. The chosen cpm.4.CSE framework was then adapted and modified for primary school context.
   c. The study of existing CMS was conducted to better fit the needs of the educational platforms used, and to get an understanding of the differences of the basic contest management principles appeared. Representatives from 32 countries filled in the prepared questionnaire. The study showed that among 19 different CMS, there is no such an educational platform where structural selection of ICDT could be directly implemented.
2. The concept-driven Informatics education model (CDIEM) aimed to extend the educational platform was created. The model is based on three main components: (1) extended cmp.4.CSE model, documented by IDEF0; It enables to perform the identification process of Informatics competencies and concepts; (2) adaptation of the two-dimensional categorization system for Informatics concept-driven tasks; (3) integration of ICDT template to CMS for well-structured selection of learning tasks.
3. The template for Informatics concept-driven tasks was developed based on 15 years of experience collected while creating and using Informatics learning tasks in 68 countries and discussed in the community. They were involved in the implementation and validation of the template. Using the designed template, annually ~150 ICDT are created and accepted by the international Bebras community.
4. The proposed CDIE model enables the extension of the educational platform where the new task selection module is integrated.

Additional ICDT template attributes are also added to CMS to facilitate a well-structured selection of ICDT for educational process.

5. The experts' evaluation of the quality in use of the developed EEP showed a very high overall quality that corresponds to 93.57 % of absolute quality, and from high to very high quality with regard to effectiveness (97.14 %), efficiency (91.43 %), flexibility (94.29 %), and context completeness (91.43 %) criteria.

FUTHER WORKS

In this thesis only the initial framework on how concept-driven Informatics education for primary school could be carried out is presented.

The cmp.4.CSE framework was chosen to apply for primary school context. Based on that, the CDIEM was developed.

The cmp.4.CSE as well as CDIE model have required a lot of work on selection of the list of Informatics concepts. The core list of concepts for primary education identified and argued in the thesis. However, for further investigations and substantiation of provided list of Informatics concepts the comprehensive pedagogical as well as psychological studies are needed.

And much more: the developed CDIE model can be extended for secondary education. Again, a new selection of the Informatics concepts need to be done in regards to pedagogical-psychological issues suitable to the age of students.

REFERENCES

1. ACM/IEEE-CS Joint Curriculum Task Force. (2001). Computing Curricula 2001 Computer Science. Retrieved from https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf
2. A Curriculum for School. (2012). Computing at School. Retrieved from http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf
3. American Heritage Dictionary of the English Language, Fifth Edition. (2011). Retrieved March 25 2019 from https://www.thefreedictionary.com/information+science
4. American Heritage Dictionary of the English Language, Fifth Edition. (2016). Retrieved May 20 2019 from https://www.thefreedictionary.com/student
5. Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Journal of Educational Technology & Society, 19(3).
6. Armoni, M. (2012). Teaching CS in Kindergarten: How Early Can the Pipeline Begin? ACM Inroads, 3(4), 18-19.
7. Association for Computing Machinery. (2003). A Model Curriculum for K-12. ACM Computer Science. New York, NY: ACM.
8. Australian Curriculum: Digital Technologies, v8.3. (2016). Available from http://www.australiancurriculum.edu.au/technologies/digital-technologies/structure
9. Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., ... & Stupurienė, G. (2015). Concepts in K-9 Computer Science Education. In Proceedings of the 2015 ITiCSE on working group reports, ACM, 85-116.
10. Barendsen, E., & Steenvoorden, T. (2016). Analyzing Conceptual Content of International Informatics Curricula for Secondary Education. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer International Publishing, 14-27.
11. Barendsen, E., Grgurina, N., & Tolboom, J. (2016). A New Informatics Curriculum for Secondary Education in The Netherlands. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer International Publishing, 105-117.

12. Barr, V., Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? Inroads, 2(1).

13. Belford, G., G. (2017). Computer Science. Encyclopedia Britannica, inc.,

14. Bell, T., Andreae, P., & Robins, A. (2014). A Case Study of the Introduction of Computer Science in NZ Schools. ACM Transactions on Computing Education (TOCE), 14(2), 10.

15. Ben-Ari, M. (2001). Constructivism in Computer Science Education. Journal of Computers in Mathematics and Science Teaching, 20(1), 45-73.

16. Biolchini, J., Mian, P. G., Natali, A. C., and Travassos G. H. (2005). Systematic Review in Software Engineering: Relevance and Utility, Technical Report ES67905, PESC – COPPE/UFRJ

17. Bischof, E., & Sabitzer, B. (2011). Computer Science in Primary Schools–Not Possible, But Necessary?! In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer, Berlin, Heidelberg, 94-105.

18. Bruner, J., S. (1960). The Process of Education (Cambridge MA, Harvard University Press).

19. Byrne, P. (1995). Fuzzy Analysis: A Vague Way of Dealing with Uncertainty in Real Estate Analysis? Journal of Property Valuation and Investment, 13(3), 22-41.

20. Byrnes, J. P., & Wasik, B. A. (1991). Role of Conceptual Knowledge in Mathematical Procedural Learning. Developmental Psychology, 5, 777-786.

21. Carey, S. (2009). The Origin of Concepts. Oxford University Press. ISBN 978-0-19-536763-8.

22. Carvalho, S., White, H. (1997). Combining the Quantitative and Qualitative Approaches to Poverty Measurement and Analysis: The Practice and the Potential. World Bank Technical Paper 366. Washington, D.C.: World Bank

23. Caspersen, M. E., & Nowack, P. (2013). Computational Thinking and Practice: A Generic Approach to Computing in Danish High Schools. In Proceedings of the Fifteenth Australasian Computing Education Conference, Australian Computer Society, Inc., 136, 137-143.

24. Chang, C. W., Wu, C. R., & Lin, H. L. (2008). Integrating Fuzzy Theory and Hierarchy Concepts to Evaluate Software Quality. Software Quality Journal, 16(2), 263-276.

25. Chaudhary, D. A. (2013). Theories of Teaching. Education, 2(3).

26. Clancey, W. J. (1995). A Tutorial on Situated Learning. Proceedings of the International Conference on Computers and Education (Taiwan). Self, J. (Ed.) Char-lottesville, VA: AACE. 49-70.

27. Cohen, H., Lefebvre, C. (Eds.). (2005). Handbook of Categorization in Cognitive Science. Elsevier.

28. Computer Science Curricula (2013). Available via internet: https://www.acm.org/education/CS2013-final-report.pdf

29. Computing Curricula. Computer Science (2001). Available via internet: http://www.acm.org/education/curric_vols/cc2001.pdf

30. Csizmadia, A. et al. (2015). Computational Thinking: A guide for teachers, available via internet: http://computingatschool.org.uk/computationalthinking

31. Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the Constructionist Learning Theory with Computational Thinking Classroom Activities. Informatics in Education, 18(1), 41-67.

32. Dagiene, V. (2005). Competition in Information Technology: An Informal Learning. In: Digital Tools for Lifelong Learning, Poland, 228-234.

33. Dagiene, V., Futschek, G. (2008). Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. In: Mittermeir, R.T., Syslo, M.M. (Eds.), Informatics Education – Supporting Computational Thinking. Lecture Notes in Computer Science. Springer, 5090, 19-30

34. Dagienė, V., Sentance, S., Stupurienė, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. Informatica, 28(1), 23-44.

35. Dagiene, V., Sentance, S. (2016). It's Computational Thinking! Bebras Tasks in the Curriculum. In: LNCS, 9973, 28-39.

36. Dagienė, V., Stupurienė, G. (2016a). Bebras - a Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. Informatics in education, 15(1), 25-44.

37. Dagienė, V., Stupurienė, G. (2016b). Informatics Concepts and Computational Thinking in K-12 Education: A Lithuanian Perspective.

Journal of information processing. Tokyo: Information Processing Society of Japan, 24(6), 732-739.

38. Dagiene, V. (2006). Information Technology Contests – Introduction to Computer Science in an Attractive Way. Informatics in Education, 5(1), 37-46.

39. Dagienė, V. (2010). Sustaining Informatics Education by Contests. LNCS, 5941, 1-12.

40. Dagienė, V., Pėlikis, E., Stupurienė, G. (2015a). Introducing Computational Thinking through a Contest on Informatics: Problem-solving and Gender Issues. Informacijos mokslai, 73, 43-51.

41. Dagienė, V., Pėlikis, E., & Stupurienė, G. (2015b). Informatinio mąstymo ugdymo užduotys: merginų ir vaikinų sprendimų analizė. Acta paedagogica Vilensia, 35, 53-66.

42. Dagienė, V., Stupurienė, G. (2014). Informatics Education based on Solving Attractive Tasks through a Contest. In: Brinda, T; Reynolds, N.; Romeike, R. (Ed.): Proceedings of the IFIP TC3 Conference "Key Competences in Informatics and ICT", 51-62.

43. Dagienė, V., Mannila, L., Poranen, T., Rolandsson, L., Stupurienė, G. (2014). Reasoning on Children's Cognitive Skills in an Informatics Contest: Findings and Discoveries from Finland, Lithuania, and Sweden. LNCS, 8730, 66–77. ISSN 0302-974. Cham: Springer International Publishing, ISBN 978331

44. Dolgopolovas, V., Jevsikova, T., Savulionienė, L., Dagienė, V. (2015) On Evaluation of Computational Thinking of Software Engineering Novice Students. In: IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations".

45. Dagienė, V., Stupurienė, G., Vinikienė, L. (2017a). Informatics Based Tasks Development in the Bebras Contest Management System. In: International Conference on Information and Software Technologies, 466-477. Springer, Cham.

46. Dagienė, V., Stupurienė, G., Vinikienė, L. (2017b). Implementation of Dynamic Tasks on Informatics and Computational Thinking. Baltic Journal of Modern Computing, 5(3), 306.

47. Dagiene, V., Stupuriene, G., Vinikiene, L., & Zakauskas, R. (2017c). Introduction to Bebras Challenge Management: Overview and Analyses of Developed Systems. In: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, 232-243. Springer, Cham.

48. Das, S. K. (2007). Handbook of Computer Science. New Delhi, India: Dominant Publishers and Distributors.

49. Denning, P. J. (1985). The Science of Computing: What Is Computer Science? American Scientist, 73(1), 16-19.

50. Department for Education, (2013). The National Curriculum in England: Framework Document (London, DfE).

51. Diethelm, I. et al., (2012). Students, Teachers and Phenomena: Educational Reconstruction for Computer Science Education. Proceedings of the Koli Calling '12, 164-173.

52. Dorling, M., Walker, M. (2014). Computing Progression Pathways V2.0. Computing at School, Available from community.computingatschool.org.uk/resources/2324

53. Duncan, C., & Bell, T. (2015). A Pilot Computer Science and Programming Course for Primary School Students. In Proceedings of the Workshop in Primary and Secondary Computing Education, 39-48. ACM.

54. Duncan, C., Bell, T., & Atlas, J. (2017). What Do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum. In Proceedings of the Nineteenth Australasian Computing Education Conference, 65-74. ACM.

55. El-Sharef, B., & El-Kilany, K. S. (2011). Process Modeling and Analysis of a Quality Management System for Higher Education. In Proceedings of the World Congress on Engineering, 1, 6-8.

56. Encyclopedia of Information Science and Technology (internet), (2018). Available: https://www.igi-global.com/dictionary/educational-platform/42260

57. Encyclopedia of Philosophy, (2017). Available: http://www.iep.utm.edu/th-th-co/

58. Entwistle, N. (1981). Styles of Learning and Teaching, John Wiley, Chichester.

59. Fincher, S. (2004). Computer Science Education Research. New York, NY: Routledge Falme.

60. Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. Review of Educational Research, 87(4), 834-860.

61. Fowler, M. (2004). UML distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

62. Gagne, E. D., Yekovich, C. W., & Yekovich, F. R. (1993). The Cognitive Psychology of School Learning. Allyn & Bacon.

63. Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., ... & Meyer, B. (2013). Informatics Education: Europe Cannot Afford to Miss the Boat. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education.

64. Gasperovic, J., Caplinskas, A. (2006). Methodology to Evaluate the Functionality of Specification Languages. Informatica 17(3), 325-346.

65. Gibson, J. P. (2012). Teaching Graph Algorithms to Children of All Ages. In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, 34-39. ACM.

66. Giddens, J. F., & Brady, D. P. (2007). Rescuing Nursing Education from Content Saturation: The Case for a Concept-Based Curriculum. Journal of Nursing Education, 46, 65-69.

67. von Glasersfeld, E. A. (1995) Constructivist Approach to Teaching. In L. P. Steffe & J. Gale (Eds.), Constructivism in education. Hillsdale, NJ: Lawrence Erlbaum Associates, 3-15.

68. Gonzalez, R., & Dahanayake, A. (2007). A Concept Map of Information Systems Research Approaches. In: Proceedings of the 2007, IRMA International Conference, Vancouver.

69. Gredler, M. E. (2008). Learning and Instruction. Upper Saddle River, NJ: Pearson Prentice Hall.

70. Grover, S., Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38-43.

71. Grillenberger, A., Przybylla, M., & Romeike, R. (2016). Bringing CS Innovations to the Classroom: A Process Model of Educational Reconstruction. ISSEP 2016, 31-39.

72. Grillenberger, A., & Romeike, R. (2017). Key Concepts of Data Management: An Empirical Approach. In Proceedings of the 17th Koli Calling Conference on Computing Education Research, 30-39. ACM.

73. Grundspenkis, J., & Strautmane, M. (2009). Usage of Graph Patterns for Knowledge Assessment Based on Concept Maps. Scientific Journal of Riga Technical University. Computer Sciences, 38(38), 60-71.

74. Gudavičius, A. (2007). Gretinamoji semantika. Šiauliai: ŠU leidykla

75. Gudavičius A. (2009). Etnolingvistika (Tauta kalboje). Šiauliai.

76. Gudavičius, A. (2011). Reikšmė – sąvoka – konceptas ir prasmė. Res Humanitariae, 10(2), 108-119.

77. Hadjerrouit, S. (2009). Teaching and Learning School Informatics: A Concept-Based Pedagogical Approach. Informatics in Education, 8(2), 227-250.

78. Hager, P. J., Scheiber, H. J., & Corbin, N. C. (1997). Designing & Delivering: Scientific, Technical, and Managerial Presentations. John Wiley & Sons.

79. Haseski, H.I., Ilic, U., Tugtekin, U. (2018). Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. International Education Studies, 11(4), 29.

80. Heintz, F., Mannila, L., & Färnqvist, T. (2016). A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K-12 Education. In: Frontiers in Education Conference (FIE), IEEE, 1-9.

81. Hromkovic, J., & Lacher, R. (2017). How to Convince Teachers to Teach Computer Science Even If Informatics Was Never a Part of Their Own Studies. Bulletin of EATCS, 3(123).

82. Hromkovič, J. (2006). Contributing to General Education by Teaching Informatics. In: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, Springer, Berlin, Heidelberg, 25-37.

83. Hounsell, D. (1997). Understanding Teaching and Teaching for Understanding. In: Marton, F., Hounsell, D. and Entwistle, N. (Eds). The Experience of Learning, Scottish Academic Press, Edinburgh, 238-57.

84. Hubwieser, P. (2007). Didaktik der Informatik [Computer science education]. Berlin, Germany: Springer.

85. Hu, C. F., Wu, C. C., Lin, Y. T., & Wang, A. T. (2017). How Computer Scientists and Computing Teachers Think Differently in the Concepts to be Included in a Secondary School Computing Curriculum. Siu-cheung KONG The Education University of Hong Kong, Hong Kong, 50.

86. IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0. (1998). Institute of Electrical and Electronics Engineers, Inc.

87. IEEE Standard for Learning Object Metadata (2002). 1484.12.1-2002. ISBN: 0-7381-3297-7. https://ieeexplore.ieee.org/document/1032843

88. ISO/IEC 25010:2011. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and software quality models

89. International Society for Technology in Education & Computer Science Teachers Association (2011). Operational definition of computational thinking for K–12 educations. Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

90. Jacob, E. K. (2004). Classification and Categorization: A difference that Makes a Difference. In Library Trends.

91. Juškevičienė, A., & Dagienė, V. (2018). Computational Thinking Relationship with Digital Competence. Informatics in Education, 17(2).

92. K-12 Computer Science Framework Steering Committee. (2016). K–12 Computer Science Framework. Retrieved from http://www.k12cs.org

93. Kalas, I. Tomcsanyiova, M. (2009). Students' Attitude to Programming in Modern Informatics. Informática na Educa-ção: teoria & prática, Porto Alegre, 12(1), 127-135.

94. Kalelioglu, K. Gülbahar, Y. & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, 4(3), 583–596.

95. Kalelioglu, F., Gulbahar, Y., Madran, O. (2015). A Snapshot of the First Implementation of Bebras International Informatics Contest in Turkey. In: Brodnik, A., Vahrenhold, J. (eds.) ISSEP 2015. Springer. LNCS, 9378, 131-140.

96. Kan, S. H. (2002). Metrics and Models in Software Quality Engineering. Addison-Wesley Longman Publishing Co., Inc.

97. Kazakoff, E. and Bers, M. (2012). Programming in a Robotics Context in the Kindergarten Classroom: The Impact on Sequencing Skills. Journal of Educational Multimedia and Hypermedia, 21(4), 371-391.

98. Kazakoff, E., Sullivan, A. and Bers, M.U. (2013). The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. Early Childhood Education Journal, 41(4), 245-255.

99. Kattmann, U. et al. (1996). Educational Reconstruction – Bringing together Issues of scientific clarification and students' conceptions. Annual Meeting of the National Association of Research in Science Teaching (NARST). St. Louis.

100. Kinnunen, P. (2009). Challenges of Teaching and Studying Programming at a University of Technology-Viewpoints of Students, Teachers and the University. Doctoral dissertation, Helsinki University of Technology.

101. Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004). Evidence-based Software Engineering. In Software Engineering, ICSE 2004, 273-281.

102. Koeppen, K., Hartig, J., Klieme, E., & Leutner, D. (2008). Current Issues in Competence Modeling and Assessment. Journal of Psychology, 216(2), 61-73.

103. Krippendorff, K. (2004). Reliability in Content Analysis: Some Common Misconceptions and Recommendations. Human communication research, 30(3), 411-433.

104. Kristan, N., Gostisa, D., Fele-Zorz, G., Brodnik, A. (2014). A High-availability Bebras Competition System. In: Gulbahar, Y., Karata, E. (eds.) ISSEP 2014. Springer, Heidelberg, LNCS, 8730, 78-87.

105. Ku, W. A. (2007). Using Concept Maps to Explore the Conceptual Knowledge of Technology Students: An Exploratory Study. Doctoral dissertation, The Ohio State University.

106. Kurilovas, E., & Serikoviene, S. (2013). New MCEQLS TFN Method for Evaluating Quality and Reusability of Learning Objects. Technological and Economic Development of Economy, 19(4), 706-723.

107. Lakoff, G. (1987). Women, Fire and Dangerous Things. The University of Chicago Press.

108. Lee, I., Martin, F., Apone, K. (2014). Integrating Computational Thinking Across the K-8 Curriculum. ACM Inroads, 5(4), 64-71.

109. Leff, A., & Rayfield, J. T. (2001). Web-application Development Using the Model/view/controller Design Pattern. In: Proceedings of Enterprise Distributed Object Computing Conference, EDOC'01. Fifth IEEE International, 118-127.

110. Lhotska, L., Bursa, M., Huptych, M., Chudacek, V., & Havlik, J. (2013). Interoperability of Medical Devices and Information Systems. In Handbook of Research on ICTs for Human-Centered Healthcare and Social Care Services, IGI Global, 749-762.

111. Lu, J. J. & Fletcher, G.H. (2009). Thinking about Computational Thinking. ACM SIGCSE Bulletin, 41(1), 260-264.

112. Loui, M. C. (1987). Computer Science Is an Engineering Discipline. Engineering Education, 78(3), 175-78.

113. Machanick, P. (2007). A Social Construction Approach to Computer Science Education. Computer Science Education, 17(1), 1-20.

114. Manev, K., & Maneva, N. (2017). On a Methodology for Creating School Curricula in Computing. Olympiads in Informatics, 11, 93-107.

115. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., Settle, A. (2014). Computational Thinking in K-9 Education. In: Goldweber, M. (ed.): ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education, 1-29.

116. Margolis, J., & Fisher, A. (2003). Unlocking the Clubhouse: Women in computing. MIT press.

117. Marton, F., & Saljo, R. (1997). Approaches to Learning. Eds. Marton, F., Hounsell, D. and Entwistle, N. The Experience of Learning.

118. Menzel, C., & Mayer, R. J. (1998). The IDEF Family of Languages. In Handbook on architectures of information systems, Springer Berlin Heidelberg, 209-241.

119. Mills, S. (2016). Conceptual Understanding: A Concept Analysis. The Qualitative Report, 21(3), 546.

120. Mittermeir, R., Bischof, E., Hodnigg, K. (2010). Showing Core-Concepts of Informatics to Kids and Their Teachers. LNCS, Springer, 5941, 143-154.

121. Morgado, E. M. M., Ortuño, R. A. C., Yang, L. L., & Ferreras-Fernández, T. (2018). Adaptation of Descriptive Metadata for Managing Educational Resources in the GREDOS Repository. In Online Course Management: Concepts, Methodologies, Tools, and Applications, IGI Global, 2063-2085.

122. Mühling, A. M. (2014). Investigating Knowledge Structures in Computer Science Education. Doctoral dissertation, Technische Universität München.

123. Murphy, G. (2002). The Big Book of Concepts. Massachusetts Institute of Technology. ISBN 0-262-13409-8.

124. Novak, J., D., Cañas, A., J. (2008). The Theory Underlying Concept Maps and How to Construct and Use Them. Technical Report. Institute for Human and Machine Cognition, Pensacola.

125. Opmanis, M., Dagiene, V., Truu, A. (2006). Task Types at Beaver Contests. Information Technologies at School, 509-519.

126. Oppermann, R., & Reiterer, H. (1997). Software Evaluation Using the 9241 Evaluator. Behaviour & Information Technology, 16(4-5), 232-245.

127. Papert, S. (1987). Computer Criticism vs. Technocentric Thinking. Educational Researcher, 16(1), 22-30.

128. Papert, S. (1996). A Word for Learning. In: Kafai, Y., Resnick, M. (eds.) Constructionism in Practice, Lawrence Erlbaum associates Inc., New Jersey, 9-24.

129. Papaurėlytė, S. (2002). Liūdesio konceptualizavimas lietuvių kalboje. Kalbotyra, (51), 1.

130. Papaurelytė-Klovienė, S. (2004). Liūdesio konceptas lietuvių ir rusų kalbose. Daktaro disertacija. Vilnius.

131. Papaurelytė-Klovienė, S. (2005). Konceptualusis lūdesio modelis lietuvių ir rusų kalbų pasaulėvaizdžiuose. Žmogus kalbos erdvėje.

132. Papaurelytė-Klovienė, S. (2007). Lingvistinės kultūrologijos bruožai. Šiauliai

133. Parmaxi, A., & Zaphiris, P. (2014). The Evolvement of Constructionism: An Overview of the Literature. In International Conference on Learning and Collaboration Technologies, Springer, Cham, 452-461.

134. Piaget, J. (1971). Psychology and Epistemology. New York: Grossman.

135. Piaget, J. (1953). The Origin of Intelligence in the Child. London: Routledge.

136. Robinson, R., Molenda, M., & Rezabek, L. (2008). Facilitating Learning. Educational technology: A definition with commentary. Routledge, 15-48.

137. Rychen, D. S., & Salganik, L. H. (2003). Definition and Selection of Competencies: Theoretical and Conceptual Foundations (DeSeCo). Summary of the final report: Key Competencies for a Successful Life and a Well-Functioning Society.

138. Schwill, A. (1997). Computer Science Education Based on Fundamental Ideas. In: Proceedings of the IFIP TC3 WG3.1/3.5 Joint Working Conference on Information Technology: Supporting Change through Teacher Education, 285-291.

139. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., Boucher-Owens, B., Stephenson, C., Verno, A. (2011).

CSTA K–12 Computer Science Standards. Computer Science Teachers Association and Association for Computing Machinery.

140. Selby, C. (2014). How Can the Teaching of Programming Be Used to Enhance Computational Thinking Skills? Doctoral dissertation.

141. Selby, C., Woollard, J. (2013). Computational Thinking: The Developing Definition. In, Special Interest Group on Computer Science Education (SIGCSE) 2014, Atlanta, US

142. Selby, C., Dorling, M., Woollard, J. (2014). Evidence of Assessing Computational Thinking. University of Southampton, 1-11.

143. Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a Phenomenology of Computational Thinking in STEM Education. arXiv preprint arXiv:1801.09258.

144. Sutherland, R. (2013). Education and Social Justice in a Digital Age. Policy Press.

145. Skūpienė, J. (2010). Evaluation of Algorithm-code Complexes in Informatics Contests. Doctoral dissertation, PhD thesis, Matematikos ir informatikos institutas.

146. Syllabus di Elementi di Informatica la scuola dell'obbligo, 2010. Retrieved from https://www.olimpiadiproblemsolving.it/documenti/SYLLABUS.pdf

147. Sysło, M. (2017). Implementing Computer Science Curriculum in Schools in Poland: Issues, Challenges, and Practice, WCCE, Ireland.

148. Strawhacker, A., Portelance, D., Lee, M. and Bers, M.U. (2015). Designing Tools for Developing Minds: The Role of Child Development in Educational Technology. Proceedings of the 14th International Conference on Interaction Design and Children (IDC'15), ACM, Boston, MA, USA.

149. Stupurienė, G.; Vinikienė, L.; Dagienė, V. (2016). Students' Success in the Bebras Challenge in Lithuania: Focus on a Long-Term Participation. ISSEP 2016. Proceedings. LNCS, 9973. Cham: Springer International Publishing, 78-89.

150. Sysło, M. M., & Kwiatkowska, A. B. (2015). Introducing a New Computer Science Curriculum for All School Levels in Poland. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer International Publishing, 141-154.

151. Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2003, 2006). A Model Curriculum for K–12 Computer

Science: Report of the ACM K–12 Task Force Computer Science Curriculum Committee, New York, NY: Association for Computing Machinery.

152. Tucker, A. B. (Ed.). (2004). CRC Handbook of Computer Science and Engineering (2nd ed.). Boca Raton, FL: CRC Press.

153. Zendler, A., Spannagel, C. (2008). Empirical Foundation of Central Concepts for Computer Science Education. ACM Journal on Educational Resources in Computing, 8(2).

154. Zendler, A., Seitz, C., & Klaudt, D. (2016). Process-Based Development of Competence Models to Computer Science Education. Journal of Educational Computing Research, 54(4), 563-592.

155. Zendler, A., Spannagel, C., & Klaudt, D. (2011). Marrying Content and Process in Computer Science Education. IEEE Transactions on Education, 54(3), 387-397.

156. Zendler, A., Klaudt, D., & Seitz, C. (2014). Empirical Determination of Competence Areas to Computer Science Education. Journal of Educational Computing Research, 51(1), 71-89.

157. Vaníček, J. (2013). Introducing Topics from Informatics into Primary School Curricula: How Do Teachers Take It? LNCS, 7780, 41.

158. Vaníček, J. (2014). Bebras Informatics Contest: Criteria for Good Tasks Revised. LNCS, 8730, 17-28.

159. Verno, A., Carter, D., Cutler, R., Hutton, M., Pitt, L. (2006). A Model Curriculum for K-12 Computer Science Level 2 Objectives and Outlines. SIGCSE05, St. Louis, Missouri, USA.

160. Walsh, J. (2011). Information Literacy Instruction: Selecting an Effective Model. Elsevier.

161. Warburton, K. (2003). Deep Learning and Education for Sustainability. International Journal of Sustainability in Higher Education, 4(1), 44-56.

162. Webb, M. E., Bell, T., Davis, N., Katz, Y. J., Fluck, A., Sysło, M. M., ... & Brinda, T. (2018). Tensions in Specifying Computing Curricula for K-12: Towards a Principled Approach for Objectives. IT-Information Technology, 60(2), 59-68.

163. Wethington, E., & McDarby, M. L. (2015). Interview methods (Structured, Semistructured, Unstructured). The Encyclopedia of Adulthood and Aging, 1-5.

164. Wiley, D., A. (2000). Connecting Learning Objects to Instructional Design Theory: A Definition, A Metaphor, and A Taxonomy, in Wiley,

David A. (DOC), The Instructional Use of Learning Objects: Online Version.

165. Wing, J. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

166. Wing, J. (2014). Computational Thinking Benefits Society. 40th Anniversary Blog of Social Issues in Computing.

167. Wilensky, U., (1991). Abstract Meditations on the Concrete and Concrete Implictions for Mathematics Education. In: Harel, I., Papert, S. (eds.) Constructionism, Ablex Publishing Corporation, Norwood, 193-203.

168. Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational Thinking and Expository Writing in the Middle School. ACM Transactions on Computing Education (TOCE), 11(2).

169. Yang, S., & Park, S. (2014). Teaching Some Informatics Concepts Using Formal System. Informatics in Education, 13(2), 323.

Gabrielė Stupurienė

CONCEPT-DRIVEN INFORMATICS EDUCATION: EXTENSION OF COMPUTATIONAL THINKING TASKS AND EDUCATIONAL PLATFORM FOR PRIMARY SCHOOL

Doctoral Dissertation
Technological Sciences, Informatics Engineering T 007

Editor Zuzana Šiušaitė

UŽRAŠAMS

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt,
www.leidykla.vu.lt
Tiražas 20 egz.