Informatics in Education, 2013, Vol. 12, No. 1, 125–142

Teaching of Computer Science Topics Using Meta-Programming-Based GLOs and LEGO Robots

Vytautas ŠTUIKYS, Renata BURBAITĖ, Robertas DAMAŠEVIČIUS
Kaunas University of Technology, Software Engineering Department
Studentų 50-415, 51368 Kaunas, Lithuania
e-mail: {vytautas.stuikys, renata.burbaite, robertas.damasevicius}@ktu.lt

Received: December 2012

Abstract. The paper's contribution is a methodology that *integrates* two educational technologies (GLO and LEGO robot) to teach Computer Science (CS) topics at the school level. We present the methodology as a framework of 5 components (pedagogical activities, technology driven processes, tools, knowledge transfer actors, and pedagogical outcomes) and interactions among the components. GLOs are meta-programmed entities to generate LO instances on demand depending on the context of use and learning objectives. A GLO is a black-box entity, which is integrated in the framework through the generating process to source the teaching and learning process via robot-based visualization to demonstrate how programs and algorithms are transformed into real-world tasks and processes. The methodology is tested in the real e-learning setting. The pedagogical outcomes are evaluated by empirical data showing the increase of student engagement level, higher flexibility and reuse enhancement in learning.

Keywords: learning object (LO), generative learning object (GLO), LEGO NXT robot, CS teaching, educational visualization.

1. Introduction

Today, Computer Science (shortly CS) is regarded as the fundamental course (similarly to mathematics, physics), which is delivered in both universities and schools. Its importance has been recognized far ago because it is a source of the primary and fundamental knowledge needed for ours life and activities, which are highly penetrated by the use of computers, Internet and other modern technologies. On the other hand, CS can be also seen as an interdisciplinary course, for example, with respect to its relation to robotics and e-learning domains. The learning and teaching processes within the e-learning environments are guided and underpinned by two basic components – pedagogy and technology – both being fuelled by teaching content (or CS teaching material in our case) that in the scientific literature is also known as learning object (LO) or learning objects (LOs).

Research on LOs forms a separate topic in e-learning domain (see, for example, Northrup, 2007). Among multiple ideas and approaches proposed and dealt with in this branch of research, the generative learning objects (GLOs) should be mentioned in the

first place. Boyle *et al.* (2004), Morales *et al.* (2005) have introduced the GLO concept and approaches based on it aiming to enforce the reuse potential in the e-learning domain. Here, the term 'generative' should be understood as a property of the learning content to be produced and handled either semi-automatically or automatically under support of some technology. The contribution of GLOs in e-learning is that the extremely wide community involved in learning has received a sign to move from the *component-based reuse model* (it relates to the use of LOs) to the *generative-based reuse model*, which relates to the use of GLOs.

The aim of this paper is to present a framework of teaching CS topics in the e-learning setting using GLOs combined with robot-based environment at the secondary school (Balčikonis gymnasium, Panevėžys). We have selected the LEGO NXT robot as the e-learning environment because of its popularity to introduce constructivist-based learning through the use of problem-based or project-based models in teaching CS topics.

The contribution of the paper is twofold: (1) the systematic approach described as a framework, were two technologies (GLO and robot programming) are seamlessly integrated; (2) a case study with the identified measurable pedagogical outcomes such as flexibility for teachers and students (due to feedback within the processes and activities of the framework), the increased students' engagement level in learning, etc.

The paper is organized as follows. Section 2 analyzes the related work. Section 3 provides a general description of the approach, and subsections of Section 3 deliver some details of the components of the approach. Section 4 presents and analyses a case study on how the approach is implemented in real e-learning setting. Section 5 evaluates the results from teacher's and student's perspectives. Finally, Section 6 presents conclusion.

2. Related Work

CS deals with abstract topics and most secondary school students have difficulties to understand and use basic concepts, such as data structures and algorithms, to create programs that solve concrete problems. The following papers emphasize the importance of at least two items in learning and teaching: (a) choosing of the relevant theory and model, educational methods, activities and environments; (b) an adequate level of student engagement into the process (Fagin *et al.*, 2001; Lubitz, 2007; Pásztor *et al.*, 2010; Pears, 2010; Hazzan *et al.*, 2011; Cowden *et al.*, 2012).

Usually the learning theory is introduced through educational methods, activities and environments. There are three main categories of learning theories: behaviorism, cognitivism and constructivism (Leonard, 2002; Smith, 2003). Behaviorism is based on using an educational environment, which forms appropriate student's behaviour and correct responses. The reinforcement of behaviour is a central issue in learning process. According to the cognitivism, the student is an active goal-oriented information receiver, processor and developer of new information, and information processing is more important than the final result. The main idea of constructivism is that the student constructs own knowledge based on his previous knowledge, own experience and learning context. According

to this approach, the main task of the teacher is to create a learning environment in which the students could actualize previous knowledge and experience and could adopt new information actively.

The constructivist-based approach dominates in CS teaching and learning (Ben-Ari, 1998; Pásztor *et al.*, 2010; Pears, 2010; Hazzan *et al.*, 2011). The approach highlights that "in this situation the students stand in the centre of learning process and the teacher only helps, gives advises as a facilitator". Jenkins (2001) indicates that the teaching environments, learning activities and teaching methods have a significant impact on motivation. If the above listed items are chosen properly, the students can learn CS topics in the most effective way.

Educational robots offer new benefits to implementing the most effective active learning methods and supporting tools for teaching of CS topics (Fagin and Merkle, 2002; Alimisis *et al.*, 2007; Frangou *et al.*, 2008; Kurebayashi *et al.*, 2008). In this context, the most commonly used learning methods derived from the constructivism-based theory are as follows: problem-based learning (Mosley and Kline, 2006; Turner and Hill, 2007; Adams *et al.*, 2010; Castledine and Chalmers, 2011; Lin and Liu, 2012), project-based learning (Sucar *et al.*, 2005; Arlegui *et al.*, 2011; Janiszek *et al.*, 2011) and game-based learning (Atmatzidou *et al.*, 2008; Lye *et al.*, 2011; Hamada and Sato, 2011). Learner-centred robotic enhanced environments based on the constructivist approach and a methodology to involve students to knowledge construction are described in (Gerndt and Lüssem, 2011; Grabowski and Brazier, 2011; Burbaite *et al.*, 2012; Petrovič and Balogh, 2012).

Some researchers (Fagin and Merkle, 2002; Weingarten *et al.*, 2007; Kim and Jeon, 2009) define the CS content in different levels of education (primary school, secondary school, university), which can be learned by students using robot-based environments. The entire CS course can be covered and robotics-based curriculum constructed using robot-enhanced environments (Sklar *et al.*, 2007). According to Adams *et al.* (2010), "the process skills have been introduced and mediated by the use of reusable learning objects (RLOs) within a virtual learning environment. Separate RLOs have also been used to develop skills in using the robots".

Analysis of component-based reusability of LO (the latter is defined by the IEEE standard as "*any digital entity, which can be used, reused or referenced during technology-supported learning*") is beyond the scope of this paper. The generative reuse model we focus here should be understood through the concept of GLO. Though the GLO-based technology in e-learning is not yet matured enough, two research directions are clearly visible already now. The first includes contributions of the pioneers of the GLO concept (Boyle *et al.*, 2004, 2008; Morales *et al.*, 2005). Their research focuses mainly on using the template-based generative technology to implement GLOs within the environment GLO Maker (`www.glomaker.org`) to teach medical-related or other courses. The second trend includes meta-programming-based GLOs (Štuikys and Damaševičius, 2007, 2008; Štuikys and Brauklytė, 2009; Štuikys and Burbaitė, 2012), which are more relevant to teach CS or some engineering courses. See relevant knowledge on meta-programming in (Štuikys and Damaševičius, 2013). Han and Krämer (2009), Oldfield (2008) can be also regarded as proponents of the GLO-based approach.

Our research on GLOs is different, as compared to other approaches, because we use meta-programming as a generative technology to implement GLOs. Despite of the effort and contribution of proponents to use the GLO-based approaches, however, this research trend is still in its infancy as compared with LO research in general due to the following reasons. The first is a short time for maturity of the approach. The second is the methodological difficulty to understand the capabilities of the approach per se. The third is the technological difficulty to acquire knowledge on which basis GLOs are to be devised because there is no uniform generative technology to design and implement GLOs. Finally, the non-technical (i.e., social) aspects are a major factor that hinders population of the approach. So far, the meta-programming-based GLOs were studied mainly within the PC-based e-learning environments. As, in fact, the GLOs are independent upon the learning tools used, we are able to combine the use of the GLOs with the robot-based environment. The remaining part of the paper deals with our new approach.

3. Description of CS Teaching Framework

Figure 1 outlines 5 basic components of the framework and their interaction. These components can be identified, for example, at the very abstract level as *pedagogy-driven activities*, *technology-driven processes, knowledge transfer channels with actors* involved, a set of *tools* used (they can also be identified as a technology), and the *pedagogical outcome*. The latter is a final product that implements the learning goal through the use of the framework in the real e-learning and teaching setting (in our case in different classes at the gymnasium level to teach CS topics). Similarly to any other product, the achieved pedagogical outcome is to be assessed. We anticipate three forms of the assessment: students' self-assessment, teacher's and expert's assessments. The framework is *generic enough* (though we do not exclude the possibility of its further generalization) because of (1) it specifies two alternative modes of using the LEGO NXT environments: with GLO or with LO introduced externally and (2) GLO can be viewed as generic LO (in other words GLO is a set of LOs, thus the use of GLOs covers the case of using LOs independently).

The framework integrates the components in some well-structured manner through relationships, knowledge transfer channels, feedback relations and decision-making procedures (see Fig. 1). The *teaching process* starts at stage 1 (task selecting) inspired by the pedagogy-driven activities, while the *learning process* starts at stage 4 (parameter selection to derive an LO from the GLO on demand to fuel the robot's actions for teaching and learning). The learning process comprises all subsequent stages and pedagogy-based activities with multiple feedback links (from *FB1* to *FB4*) as it is depicted within the framework.

Two underlying concepts (GLOs and LEGO NXT robot), on which our approach is based, are seen within the framework only through the processes they provide. Technological aspects of the LEGO NXT robot (in terms on how they are related and integrated to our teaching setting) can be found in (Burbaite *et al.*, 2012), or they can be learnt from other papers (Gerndt and Lüssem, 2011; Grabowski and Brazier, 2011; Petrovič and Balogh, 2012). In order to convince reader in soundness of the meta-programming-based GLOs, we provide a methodological background of the concept in the next subsection.

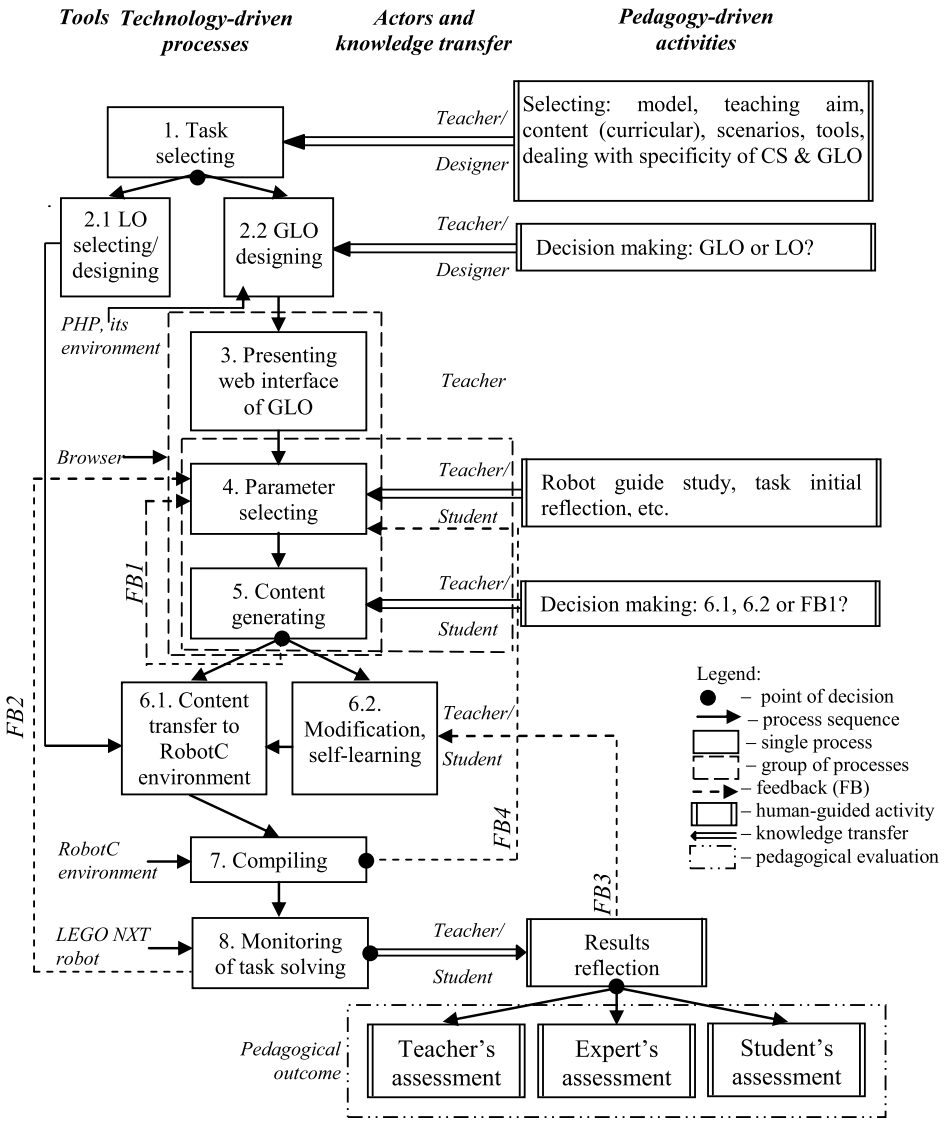


Fig. 1. CS teaching framework based on using LEGO robots and GLOs.

3.1. *Learning Variability Concept – the Background to Understand GLOs*

For CS teachers and students, a GLO is a higher-level program (otherwise meta-program), because it generates other programs (meaning lower-level ones) automatically. In contrast to pioneers of the GLO-based approach (Boyle *et al.*, 2004; Morales *et al.*, 2005), who use the template-based technology to implement GLOs, we use heterogeneous meta-programming (Štuikys and Damaševičius, 2013) as generative technology. Though there are different interpretations of the paradigm, in the most general case, meta-programming

can be defined as the technology for implementing meta-programs (otherwise program generators) through the explicit representation of domain task *variability*.

For less knowledgeable readers (because GLOs need still to live up their promises to become more widely accepted in the e-learning domain), we explain the variability concept as a basis to understand meta-programming-based GLOs more thoroughly. Let us have a very simple object, such as the linear equation ($y = ax + b$), and interpret it from the variability perspective in different domains (mathematics, CS, and e-learning).

In mathematics, for example, the equation is a canonical form meaning the general representation with the explicit statement of eligible values for the argument (x), function (y) and coefficients (a, b) as follows: $x, y, a, b \in \mathbb{R}$. In a particular case, however, some specific variants ($a > 0,\ a < 0,\ a = 0,\ b \neq 0$, etc.) may be excluded and considered separately. All these are variability in that domain, though this term is not exploited here. The function is formally defined as a mapping of the argument (variable) eligible domain D onto the function value space R, that is: $f\colon D \longrightarrow R$. The elegancy and beautifulness of the mathematics language is its potential to express the items (objects, categories) uniformly and as general and short representations as possible.

In CS (programming), the equation can be easily transformed into a computer program to calculate y values for the predetermined space of values for x, a, and b. The space in this case, however, is much narrower *as compared to the mathematical representation* due to limited computational resources. Variability (if one realizes the program) could be seen in part explicitly and in part implicitly within the program source code. Now we explain the difference between the program and meta-program using some rationale and the same example. From the outcome (i.e., program execution) perspective, a program (if it is correct and terminates) always returns a *concrete value* as a result of calculation (e.g., $y = 2$ when $a = b = x = 1$), while meta-program returns the *other program (programs)* as a value (e.g., $y := x + 1$; when both higher-level parameters a' and b' are equal to 1). This subtle difference opens the way to extend reusability and generate the program instance on demand and use it as a subject for reuse (use-as-is, transfer for other contexts with or without modification).

E-learning is not a homogeneous domain; rather it is a combination of the following sub-domains: pedagogy, information sciences, IT-based technology, sociology, and psychology and computer science (sometimes also called informatics, actually it is also a combination of the others). As a consequence, the e-learning variability (*LV*) is also not a homogeneous item; rather it is a set of the following constituents: pedagogical variability (*PV*), social variability (*SV*), content variability (*CV*) and technology variability (*TV*) and Interaction variability (*IV*) among the variability constituents mentioned before. Therefore, we have a simplified formula (1) to define *LV*:

$$LV = PV \circ SV \circ CV \circ TV \circ IV, \tag{1}$$

where "$\circ$" means a specific operator to integrate the items, which are very broad and in terms of SW engineering are treated as domains.

We can also explain variability with other concrete examples (*CV* has already been illustrated in this subsection). *PV* variants: {(teaching objective-1: <*concept presentation*>,

teaching objective-2: <*demonstration of program examples*>. . . .); (project-based model, problem solving model). . .}; *SV* variants: {age, gender, capabilities, previous knowledge}, etc. *IV* variants are more subtle and should be conceived of as relationships of the type *require* here. For example, the teaching objective for better engagement (variant of *PV*) *require*s visualization of the task (LO content).

Now we can present two definitions of the meta-programming-based GLO as it is understood in the context of this paper. *Semantically, GLO is the explicit representation of LV using heterogeneous meta-programming as generative technology. Structurally, GLO is a set of LO instances pre-specified into the structure to automatically generate a concrete instance on demand.*

It is not our intention to consider details of GLOs (such as the ones how to design GLOs, see, e.g., (Štuikys and Damaševičius, 2008)) here. Nevertheless, two underlying properties we need to state: (1) any constituent of variability may have a few or many variants as it was explained by the foregoing examples, (2) any constituent (its type and variants), despite of their semantic differences (e.g., pedagogical, technological, content), can be expressed uniformly through higher-level (meta) parameters, their values and various dependencies among the values. At this point to understand the GLO-based approach, we present the structural model of GLOs given in Fig. 2, which corresponds to the structural definition. The model is presented at two levels: higher-level that is independent upon the implementation (Fig. 2a) and lower-level, which is the technology-dependent (Fig. 2b). Here, the pure programming language details are missed (reader can see them in our case study).

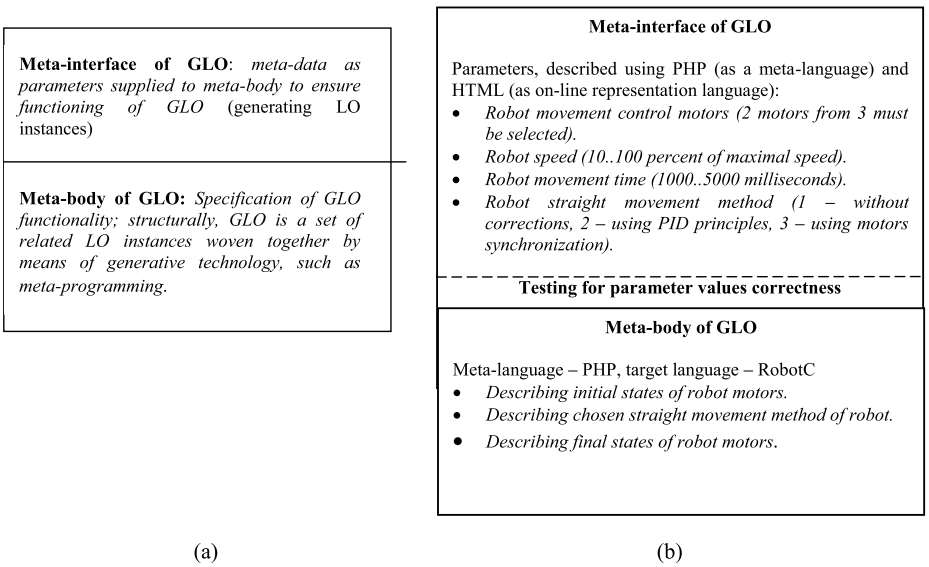


Fig. 2. (a) – General structure of GLO model, (b) – GLO model "Robot calibration (sequential algorithm)" with some details.

In summary, we formalize the model as given by formula (2):

$$\text{GLO} = \textit{Meta-interface} \times \textit{Meta-body} \tag{2}$$

(here "$\times$" means mapping).

3.2. *Pedagogy-Driven Activities*

The framework we suggest uses two learning models derived from the constructivist-based pedagogical approach: problem-based learning (Adams *et al.*, 2010; Castledine and Chalmers, 2011; Lin and Liu, 2012; Mosley and Kline, 2006; Turner and Hill, 2007) and project-based learning (Arlegui *et al.*, 2011; Sucar *et al.*, 2005; Janiszek *et al.*, 2011). Though there is a thin line among the models, nevertheless, we introduce them as slightly different teaching scenarios (in other words the models are integrated within the scenarios) either explicitly or intuitively through the learning objective formulation, teacher's plans (such as curricular content) and teaching environment (LEGO robot-based) and teaching task selection. The basic requirement for creating the scenarios is to enforce the students' involvement and engagement in the process. All these are seen as predetermined pedagogical activities before starting the teaching and learning process. One pedagogical activity, however, should be highlighted separately here, that is, the formulation of requirements for GLO design (this activity as well as design of GLOs per se are beyond the scope of this paper). Here, we accept that the teaching content (defined by the standard or enhanced CS teaching curricular program in schools) either partially or fully should be implemented as GLO or GLOs (having in mind the first mode of using the framework, see decision point at Step 2).

The other activities are clear from Fig. 1, except the final, which we consider separately in Section 5.

3.3. *Technology-Driven Processes*

Task selection. It is the first component to instantiate the other processes. It is primarily based on the knowledge that comes as a result of the pedagogy-driven activity. This knowledge should be extracted either by teacher or designer first, and only then the selection follows, though in the most general case (it is not depicted in Fig. 1), the two-side interaction may be possible. The task selection process enables to create teaching content (either as GLO or LO; here we focus on the first). It is needed because there are multiple ways to realize the teaching program. On the other hand, there are restrictions imposed by the robot-based environment and requirements for student engagement in learning. For example, the first task to be solved is to synchronize robot's characteristics to operate mechanical actions properly. Therefore the first task should be a test for the correct functioning of the robot.

From the CS perspective, it is an important aspect in which programming language, the algorithms that should cover the curricular should be selected. Another factor that affects the task selecting is the possibility of robots actions to visualize the real world problems that might appear for students as the most interesting ones (one of pedagogical

objectives is to motivate students to learn as much as possible). Examples of such problems could be: drawing ornaments or other pictures that attract students and are used in practice, to model the robot's movement in the space with obstacles, cutting some objects, such as vegetables in food preparation, etc.). On the other hand, the task selected for teaching and learning, has to cover the curricular content requirements for CS for different grade classes. Therefore, the possibilities for task selection are only one kind of sources to obtain *LV*, mainly based on content, and then to design GLOs.

Further, we miss the description of the following technology-based processes (GLO designing and on-line representing) because here we focus on using GLOs as black-box entities.

(Meta)parameter selection (process 4 in Fig. 1). In fact, the learning process starts at this stage because students (along with teacher) are involved to a larger extent. Physically, selecting parameters means the reading of the GLO meta-interface, which is a human-oriented structure being represented as graphical boxes (see for example, Fig. 3a. We define the learning process in this context as a sequence of processes 4–8 along with pedagogy-based activities underpinned with multiple feedback links (from *FB1* to *FB4*).

Content generation. The content generation (process 5 in Fig. 1) is fully automatic. The PHP processor supports this process, because the PHP has been selected as a meta-language to describe the GLO specifications. The generating process should be understood as an action of deriving LO instances on demand according to the selected parameter values. Students are able to repeat the process multiple times by selecting different values in each case. This repeatable process can be viewed as the adaptation of the generated content (we call it LO instance) to the learner's specific context. There might be such a case, when a learner is not satisfied by the generated LO instance. In this case, he/she has a possibility to change the content (perhaps with the help of teacher) manually (process 6.2). The remaining processes are not so much related with GLO itself but rather with LO instance because the job prescribed to the GLO has been already done.

Compiling and task running are well-known processes for CS courses independent upon which facilities (computer, robot, mobile phone) are used.

Feedback links (FB1-FB4) is a very important part of the learning process because they ensure a great deal of flexibility *to re-generate the content*, *to modify the content*, *to obtain knowledge* through *monitoring learning scenarios* as they are seen in the robot-based reality (but not in the virtual reality as it takes place when the only PC and Internet as learning facilities are used).

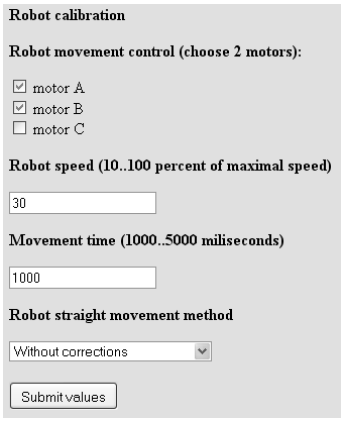
We summarize technology driven processes in combination with pedagogical activities in Table 1.

4. Case Study

We present two variants of our case study. The first illustrates the initial task as a mandatory action to prepare the robot for correct functioning. The task was implemented as GLO with 4 parameters (Fig. 3a, parameter values are given in the white boxes or marked

Table 1
Summary of the process analyzed

Item	Details
Initial requirements	Knowing of pedagogical model Knowing of curriculum content Robot readiness for the use
Guide	Human-guided, tool-guided
Automation level	Automatic, semi-automatic
Activity	Single activity, multiple reuse activity
Tools type	Hardware, software
Degree of the teacher or student involvement	Teacher/student, student/student measured by the number of FBs, visualization, adaptability (see also Table 3)
Constraints	Initial knowledge and readiness of teacher
Functionality	Described as the input/output specification
Abstraction level	How much details relevant to teaching topic should be revealed explicitly
Types of sub-processes	Robot independent, robot dependent Generative technology independent, generative technology dependent



(a)

```
task main()
{
   // Initial states of robot motors
   motor[motorC] = 50;
   wait1Msec(100);
   motor[motorC] = 0;
   // Straight movement of robot
   motor[motorA] = 30;
   motor[motorB] = 30;
   wait1Msec(1000);
   // Final states of robot motors
   motor[motorA] = 0;
   motor[motorB] = 0;
   motor[motorC] = -50;
   wait1Msec(100);
   motor[motorC] = 0;
}
```

(b)

Fig. 3. (a) – Meta-interface of GLO "Robot calibration", (b) – generated instance as LO to teach sequential algorithms/programs.

by 'v'), and its derivative instances to cover the simplest CS topics (sequential algorithms and programs, Fig. 3b). The snapshot of the learning scenarios that are observed by students is given in Figs. 3 and 4. As it was said previously, technical details of how to prepare robot mechanics for information processing is beyond of the paper's scope, though some aspects are encoded in the GLO specification as it is described by the following.

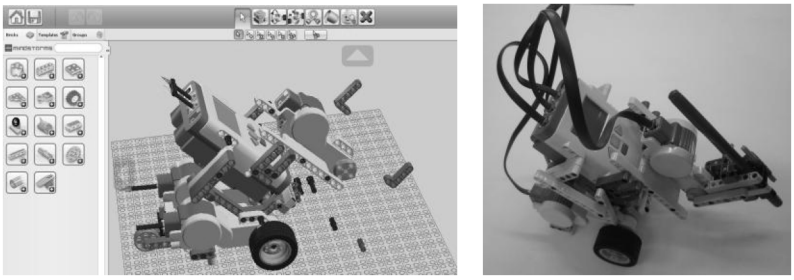
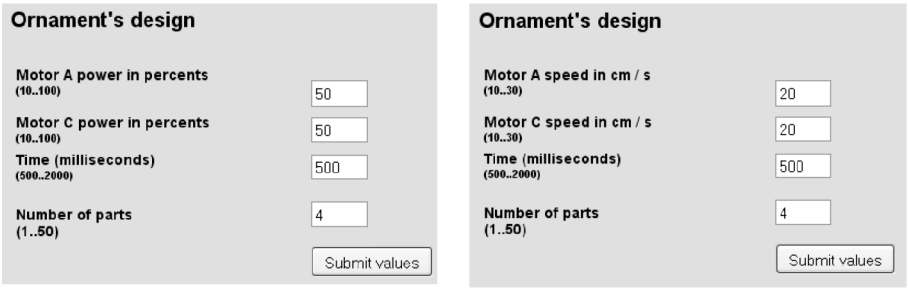


Fig. 4. (a) – DRAWBOT designing steps; (b) – a full view of DRAWBOT (Burbaitė *et al.*, 2012).



(a) (b)

Fig. 5. Meta-interface variants of the GLO "Ornaments' drawing by DRAWBOT": (a) motors speed described as power in percents, (b) motors speed given in cm/s.

Motors are controlled by specifying the needed power level to be applied to the motor. For this purpose, the programming language RobotC uses the parameter named "Power level". In Fig. 5a, the parameter is expressed in percents, while in Fig. 5b – in speed (cm/s). That is done because some students better perceive the first measure, while others – the second. Power levels range from -100 to $+100$. Negative values indicate the reverse direction and positive values indicate the forward direction. For example, to move motor A forward at 30% of full power, we use the following statement: motor[motorA] $=$ 30; (see Fig. 3b).

Now we consider the second variant of our case study. It deals with the task that responds to the requirement to ensure the possibility for better students' engagement in learning. The task (to teach loops in program) is about *visualization* of the result created by the program. The program is derived from the GLO as a LO instance. Then the instance runs within the robot environment that makes drawing to realize the visualization. As, in this case, the robot was adapted to drawing activity, it was called DRAWBOT.

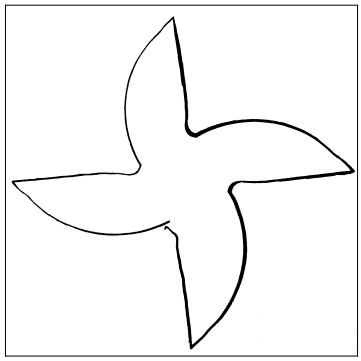
Though it is possible to apply the DRAWBOT to various tasks, we use ornament drawing here because of extremely high variability of the task (see, e.g., Table 2) and its practicality in use (e.g., ornaments can be used in artistic design). Furthermore, from the pedagogy perspective, the task enables to demonstrate the use of both the problem solving model and the project-based model in teaching. For example, Fig. 6 illustrates the use of the first model and Fig. 7 – the second.

Table 2
Generative learning objects (GLOs) to teach Computer Science topics

Number of parameters	Topic	Number of parameter	Number of LO instances possible to generate
1.	Sequential algorithms/programs	4	135
2.	Conditional statements	4	108
3.	Loops	4	1350
4.	Nested loops	7	29160

```
task main()
{
//-----------------------------------
// Preparation for painting
   motor[motorB] = 50;
   wait1Msec(100);
   motor[motorB] = 0;
   //-----------------------------------
   // Painting
   for (int j = 0; j < 4; j++) {
      motor[motorC] = 50;
      motor[motorA] = 50;
      wait1Msec(1000);
      //-------------------------------
      motor[motorC] = -50;
      motor[motorA] = 0;
      wait1Msec(1000);
   }
   //-----------------------------------
   // Painting of ornament is finished
   motor[motorB] = -50;
   wait1Msec(100);
   motor[motorB] = 0;
   //-----------------------------------
}
```

(a)



(b)

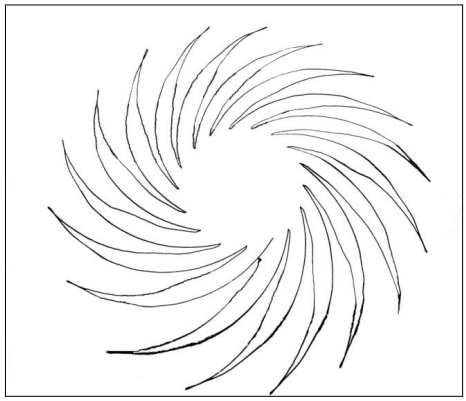
Fig. 6. (a) – Generated LO instance (from GLO) as motivating example to cover "Loop teaching", (b) – result of LO execution as a material introduced by teacher for learning at initial phase through problem solving.

Finally, we discuss the pedagogical outcome of the approach. Table 2 summarizes the SC topic teaching content which was transformed into GLO specifications by the teacher (second author of the paper). She was working as a course designer to prepare the material (GLOs) in advance. The 4th column (see Table 2) describes the content variability (*CV*, see also (1)) space. Thus, there is a wide possibility for choice to adapt the content for the use context. In practice, however, the only small part of the LO instances to fuel the robot actions are needed to use.

To evaluate the outcome through the increase of student engagement level, we have applied the methodology adapted from (Urquiza-Fuentes and Velázquez-Iturbide, 2009). Results are summarized in Table 3. Here the engagement level is categorized into 5 categories, where the first (*Viewing*) requires the least effort for students. Thus, all students have passed this level easily. Each subsequent level requires the progressively increased effort. Therefore, the number of students able to fulfil actions (as they are specified by the engagement level) is progressively decreasing. Only the small part of students were able to be evaluated by the highest engagement level (*Presenting*) in the first teaching year,

```
task main()
{
//-----------------------------------------------
// Preparation for painting
   motor[motorB] = 50;
   wait1Msec(100);
   motor[motorB] = 0;
   //-----------------------------------------------
   // Painting
   for (int j = 0; j < 17; j++) {
      motor[motorC] = 45;
      motor[motorA] = 15;
      wait1Msec(1000);
   //-----------------------------------------
      motor[motorC] = -45;
      motor[motorA] = 0;
      wait1Msec(1000);
   }
   //-----------------------------------------------
   // Painting of ornament is finished
   motor[motorB] = -50;
   wait1Msec(100);
   motor[motorB] = 0;
   //-----------------------------------------------
}
```

(a)



(b)

Fig. 7. (a) – Generated instance after corrections made by students as a result of problem solving, (b) – task solving result based on student activities.

though the number of such students was higher in the 2nd year of teaching according to the presented methodology. The results were evaluated by the teacher taking into account also the student's self-assessment.

5. Analysis and Evaluation of the Outcomes

We have discussed a methodology that integrates two technologies for learning CS topics: meta-programming-based GLOs and LEGO NXT robot-based. As the core idea on which GLOs are constructed is the *learning variability* (which comprises pedagogical, content, technological aspects such as tools), it is possible to seamlessly integrate the *process of* using GLOs within the LEGO NXT robot environment into a coherent e-learning setting. In this paper, our focus was given on the *synergy effect* of using GLOs and robots. As tools that support GLOs enable to *automatically generate LO instances on demand* for sourcing the robot functioning, the first technology *ensures flexibility* to a much larger extent as compared with the LOs prepared manually. Since we can express flexibility with the *level of automation*, by the *number of feedback links*, by *modes for adaptation and changes*, GLO-based technology extends the reuse dimension too. What is the most important that this dimension is measurable. In essence, the GLO technology is independable upon the tools used for learning (robots, PC, mobiles).

Thus, we can speak about *the enhancement of internal reusability* through the use of GLOs (in our case within the robot environment), or about *the enhancement of external variability* (meaning a much wider extent of reusability). The internal reuse impact on LEGO NXT can be understood as a better exploitation of its technical capabilities (by more frequent use, by the larger number of different information testing for robot functioning, etc.).

Table 3
Student engagement levels using GLO "Ornaments' and DRAWBOT" environment

No.	Engagement level	2011–2012			2012–2013		
		No. of students	No. of boys	No. of girls	No. of students	No. of boys	No. of girls
1.	*Viewing*: Students view the ornaments given by teacher passively and are passive LO consumers.	44 100%	33 100%	11 100%	67 100%	54 100%	13 100%
2.	*Responding*: Students use the visualization of ornaments actively as a resource for answering questions given by teacher and are active LO consumers.	34 77%	24 73%	10 91%	78%	43 80%	9 69%
3.	*Changing*: Students themselves modify ornaments by changing meta-parameter values using the pre-specified meta-interface and the tool and are LO designers.	33 75%	24 73%	9 82%	46 69%	37 69%	9 69%
4.	*Constructing*: Students construct their own ornaments introducing new meta-parameter values not anticipated by the meta-interface and are LO co-designers and testers.	17 39%	9 27%	8 73%	27 40%	23 43%	4 31%
5.	*Presenting*: Students present to the audience for discussion new ornaments and are treated as GLO co-designers.	5 11%	3 9%	2 18%	17 25%	15 28%	2 15%

All these observations are given from the perspective of understating what the methodology can bring to its users in general. Now we evaluate the pedagogical outcomes from the teacher's and student's perspective. From the teacher's viewpoint, the teaching and learning processes are more intensive since there are much more possibilities for creating various scenarios (variants how processes are carried out). Teacher has more time to observe student's activities, participate in student's reflections on teaching outcomes. Thus, the process is more effective and more easily organized. Of course, those are positive sides of the methodology; however, one needs to take into account the following assumptions as constraints: (1) GLOs are treated as black-box entities (meaning that GLOs have been designed in advance and they are correct); (2) Robot-based environment was created and tested in advance. A negative side of the methodology is the teacher's overload in the preparatory work and the need for knowledge that might be beyond the scope of a standard level. Not every CS teacher (if there is no external support) can go through such a trial.

From the student's perspective, the methodology gives understanding how the CS content is to be realized practically by solving real world tasks. This, combined together with pedagogical models, visualization of the problems, makes the teaching and learning process much more attractive – students see how abstract items (variables, data, types, loops within a program) are transformed into physical entities (robot's movement, speed, etc.). The methodology enforces the interdisciplinary vision to teaching and provides integrated knowledge for students. It also enforces the level of engagement leading to the

increase of student activeness. Though this is not true for each student, nevertheless, our experience has shown that, in average, such the impact on students is notable. Even more, some students were able to accept the role of co-designers (apprentices of a teacher) in developing GLOs and constructing the robot-based teaching environment.

6. Conclusion

1. Meta-programming-based GLO can be seen not only as a tool for sourcing e-learning environments but also as *a learning content per se* for teaching and learning CS-based courses**.**
2. As the methodology we have proposed is based on using a variety of technologies, students are able to receive the integrated knowledge and enforce the vision that CS is an interdisciplinary course.
3. The methodology can be also adapted to other (different from robots) environments and it can be easily extended for delivery of the entire CS course (not only selected topics as it may follow from our case study considered in this paper).

References

Adams, J., Kaczmarczyk, S., Picton, P., Demian, P. (2010). Problem solving and creativity in engineering: conclusions of a three year project involving reusable learning objects and robots. *Engineering Education*, 5(2), 4–17.

Alimisis, D., Moro, M., Arlegui, J., Pina, A., Frangou, S., Papanikolaou, K. (2007). Robotics&constructivism in education: the TERECoP project. In: Kalas, I. (Ed.), *Proceedings of the 11th European Logo Conference*, Bratislava, Slovakia, Comenius University, 1–11.

Arlegui, J., Pina, A., Moro M. (2011). A paradox in the constructive design of robotic projects in school. In: *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, Vienna, Austria, 29–34.

Atmatzidou, S., Markelis, I., Demetriadis, S. (2008). The use of LEGO mindstorms in elementary and secondary education: game as a way of triggering learning. *Workshop Proceedings of SIMPAR 2008*, 22–30.

Ben-Ari, M. (1998). Constructivism in computer science education. In: *Proceedings of SIGCSE'98*, Atlanta, USA, ACM, 257–261.

Boyle, T., Leeder, D., Chase, H. (2004). To boldly GLO – towards the next generation of learning objects. In: *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2004*, Chesapeake, VA, AACE, 28–33.

Boyle, T., Ljubojevic, D., Agombar, M. Baur, E. (2008). On conceptual structure of generative learning objects. In: *Proce of World Conference on Educational Multimedia, Hypermedia&Telecommunications*, AACE, Austria, Vienna, 4570–4578.

Burbaite, R., Stuikys, V., Marcinkevicius, R. (2012). The LEGO NXT robot-based e-learning environment to teach computer science topics. *Electronics and Electrical Engineering*, 18(2), 133–136.

Castledine, A., Chalmers, C. (2011). LEGO Robotics: an authentic problem-solving tool? *Design and Technology Education*, 6(3), 19–27.

Ley, C.N, Wong, W.K., Chiou, A. (2011). Framework for educational robotics: a multiphase approach to enhance user learning in a competitive arena. In: *Edutainment'11, Proceedings of the 6th International Conference on E-Learning and Games, Edutainment Technologies*, Springer-Verlag Berlin, 317–325.

Cowden, D., Ustek, D., O'Neill, A., Opavsky, E., Walker, H.M. (2012). A C-based Introductory course using robots. In: *The 43rd Technical Symposium on Computer Science Education*, 463–468.

Sucar, E.L., Noguez, J., Huesca, G. (2005). Project oriented learning for basic robotics using virtual laboratories and intelligent tutors. In: *Frontiers in Education. Proceedings of 35th Annual Conference*, S3H-12.

Fagin, B.S., Merkle, L. (2002). Quantitative analysis of the effects of robots on introductory computer science education. *Journal on Educational Resources in Computing (JERIC)*, 2(4), 1–18.

Fagin, B.S., Merkle, L.D., Eggers, T.W. (2001). *Teaching Computer Science with Robotics Using Ada/Mindstorms 2.0*, SIGAda, Bloomington, MN, USA, 73–77.

Frangou, S., Papanikolaoum, K., Aravecchia, L., Montel, L., Ionita, S., Arlegui, J., Pina, A., Menegatti, E., Moro, M., Fava, N., Monfalcon, S., Pagello, I. (2008). Representative examples of implementing educational robotics in school based on the constructivist approach. In: *SIMPAR Workshop on Teaching with Robotics: Didactic Approaches and Experiences*, Venice, Italy, 54–65.

Gerndt, R., Lüssem, J. (2011). Mixed-reality robotics – a coherent teaching framework. In: *Proceedings of 2nd International Conference on Robotics in Education (RiE)*, 193–200.

Grabowski, L.M., Brazier, P. (2011). Robots, recruitment, and retention: Broadening participation through CS0. In: *Frontiers in Education Conference (FIE)*, F4H-1–F4H-5.

Hamada, M., Sato, S. (2011). A game-based learning system for theory of computation using Lego NXT Robot. *Procedia Computer Science*, 4, 1944–1952.

Han, P., Krämer, B.J. (2009). Generating interactive learning objects from configurable samples. In: *International Conference on Mobile, Hybrid, an On-Line Learning (ELML'09)*, Cancun, 1–6.

Hazzan, O., Lapidot, T., Ragonis, N. (2011). *Guide to Teaching Computer Science – An Activity-Based Approach*. Springer-Verlag, London.

Janiszek, D., Pellier, D., Mauclair, J., Baron, G.L., Parchemal, Y. (2011). Feedback on the use of robots in project-based learning: how to involve students in interdisciplinary projects in order to increase their interest in computer science. *INTED2011 Proceedings*, 1815–1824.

Jenkins, T. (2001). *The Motivation of Students of Programming*. Thesis of Master of Science. The University of Kent.

Kurebayashi, S., Kanemune, S., Kamada, T., Kuno, Y. (2007). The effect of learning programming with autonomous robots for elementary school students. In: *11th European Logo Conference*, Comenius University Press, Bratislava, 1–9.

Leonard, D. (2002). *Learning Theories, A to Z*. Westport, Conn, Oryx Press.

Ley, C.N, Wong, W.K., Chiou, A. (2011). Framework for educational robotics: a multiphase approach to enhance user learning in a competitive arena. In: *Edutainment'11, Proceedings of the 6th International Conference on E-Learning and Games, Edutainment Technologies*, Springer-Verlag Berlin, 317–325.

Lin, C.H., Liu, E.Z.F. (2012). The effect of reflective strategies on students' problem solving in robotics learning. In: *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), IEEE Fourth International Conference*, 254–257.

Lubitz, W.D. (2007). Rethinking the first year programming course. In: *Proceedings of the Canadian Engineering Education Association*.
`http://library.queensu.ca/ojs/index.php/PCEEA /article/view/3811/3767`.

Morales, R., Leeder, D., Boyle, T. (2005). A case in the design of generative learning objects (GLOs): applied statistical methods. In: *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Chesapeake, VA, AACE, 2091–2097.

Mosley, P., Kline, R. (2006). Engaging students: a framework using LEGO robotics to teach problem solving. *Information Technology, Learning, and Performance Journal*, 21(1), 39–45.

Northrup, P.T. (2007). *Learning Objects for Instruction: Design and Evaluation*. Information Science Publishing, New York.

Oldfield, J.D. (2008). An implementation of the generative learning object model in accounting. In: *Proceedings of ASCILITE*, Melbourne.
`http://www.ascilite.org.au/conferences/melbourne08/ procs/oldfield.pdf`.

Pásztor, A., Pap-Szigeti, R., Lakatos Török, E. (2010). Effects of using model robots in the education of programming. *Informatics in Education*, 9(1), 133–140.

Pears, A.N. (2010). Enhancing student engagement in an introductory programming course. In: *40th ASEE/IEEE Frontiers in Education Conference*, Arlington, F1E1–F1E2.

Petrovič, P., Balogh, R. (2012). Deployment of remotely-accessible robotics laboratory. *International Journal of Online Engineering*, 8(2), 31–35.

Kim, H.S., Jeon, J.W. (2009). Introduction for freshmen to embedded systems using LEGO Mindstorms. *Education, IEEE Transactions*, 52(1), 99–108.

Sklar, E., Parsons, S., Azhar M.Q. (2007). Robotics across the curriculum. In: *AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, 142–147.

Smith, M.K. (2003). Learning theory. *The Encyclopedia of Informal Education*. `www.infed.org/biblio/b-learn.htm`. Last update: May 29, 2012.

Štuikys, V., Brauklytė, I. (2009). Aggregating of learning object units derived from a generative learning object. *Informatics in Education*, 8(2), 295–314.

Štuikys, V., Burbaite, R. (2012). Two-stage generative learning objects. In: Skersys, T., Butleris, R., Butkiene, R. (Eds.), *18th International Conference, ICIST*, Kaunas, Lithuania. *Proceedings Series: Communications in Computer and Information Science*, Vol. 319, Springer Berlin Heidelberg, 332–347.

Štuikys, V., Damaševičius, R. (2007). Towards knowledge-based generative learning objects. *Information Technology and Control*, 36(2), 202–212.

Štuikys, V., Damaševičius, R. (2008). Development of generative learning objects using feature diagrams and generative techniques. *Informatics in Education*, 7(2), 277–288.

Štuikys, V., Damaševičius, R. (2013). *Meta-Programming and Model-Driven Meta-Program Development: Principles, Processes and Techniques*, Springer. Springer.

Turner, S., Hill, G. (2007). Robots in problem-solving and programming. In: *8th Annual Conference of the Subject Centre for Information and Computer Sciences*, University of Southampton, UK, 82–85.

Urquiza-Fuentes, J., Velázquez-Iturbide, J.Á. (2009). Pedagogical effectiveness of engagement levels – a survey of successful experiences. *Journal Electronic Notes Theoretical Computer Science*, 224, 169–178.

Weingarten, J.D., Koditschek, D.E., Komsuoglu, H., Massey, C. (2007). Robotics as the delivery vehicle: a contextualized, social, self paced, engineering education for life-long learners. In: *Robotics Science and Systems Workshop on "Research in Robots for Education"*, 1–6.

V. Štuikys is a professor of Software Engineering Department at KUT and holds the degree of habilitate doctor of cience. His research interests include system design methodologies based on reuse and automatic program generation and transformation. He is author of about 100 papers in this area and co-author of the monograph „Meta-Programming and Model-Driven Meta-Program Development: Principles, Processes and Techniques" published by Springer. His specific interest is the application of the design methodologies to the e-learning domain. Currently he is a head of research group at KUT and supervisor of 3 PhD students. He is a member of ACM and IEEE.

R. Burbaitė is a PhD student at Software Engineering Department of Kaunas University of Technology (KUT) and a teacher of computer science (CS, Informatics) at Balčikonis Gymnasium (Panevėžys, Lithuania). She also provides teaching of students in informatics at KUT. Her research interests relate to the e-learning domain that includes the use of meta-programming-based generative learning objects and educational robots to teach CS topics. She is a member of Interest Group of Informatics Teachers.

R. Damaševičius is a professor at Software Engineering Department, KUT. He received his PhD (2005) degree in informatics engineering from KUT. Currently he teaches programming, robotics and software engineering courses. He is also the member of Design Process Automation Group at Software Engineering Department. His research interests include program transformation and meta-programming, design automation and software generation, as well as domain analysis methods. Currently he is a supervisor of 3 PhD students. He is also a member of ACM.

Informatikos mokymas(is) panaudojant generatyvinius metaprogramavimu grindžiamus mokymo(si) objektus ir LEGO robotus

Vytautas ŠTUIKYS, Renata BURBAITĖ, Robertas DAMAŠEVIČIUS

Straipsnyje pateikiama metodologija, integruojanti generatyvinių mokymo(si) objektų (GMO) ir LEGO robotų technologijas. Ši metodologija skirta informatikos mokymui(si) vidurinėje mokykloje. Ją sudaro 5 komponentai (pedagoginė veikla, technologiniai procesai, įrankiai, žinių perdavimo dalyviai ir pedagoginiai rezultatai) bei jų tarpusavio sąryšiai. GMO specifikuojami metaprogramomis, kurias vykdant generuojami mokymo(si) objektų egzemplioriai pagal poreikį, priklausomai nuo konteksto ir mokymo(si) tikslų. Mokytojo ir mokinio požiūriu, GMO yra „juodoji dėžė“, kuri integruota į pasiūlytą struktūrą naudojant generavimo procesą. Mokymo(si) procesas vykdomas LEGO robotų aplinkoje, o mokymo turinys sukuriamas iš GMO. Aplinka įgalina vizualizuoti mokymo turinį transformuojant algoritmus ir programas į realaus pasaulio uždavinius ir procesus. Metodologija išbandyta realioje e. mokymosi aplinkoje. Pedagoginiai rezultatai (mokinių motyvacijos padidėjimas, mokymo(si) medžiagos automatinis rengimas, užtikrinantis lankstumą ir pakartotinį panaudojimą) įvertinti pateikiant empirinius duomenis.