Baltic J. Modern Computing, Vol. 5 (2017), No. 2, 183-205
http://dx.doi.org/10.22364/bjmc.2017.5.2.03

Stage-Based Generative Learning Object Model for Automated Content Adaptation

Vytautas ŠTUIKYS, Renata BURBAITĖ, Kristina BESPALOVA, Tomas BLAŽAUSKAS, Dominykas BARISAS

Faculty of Informatics, Kaunas University of Technology, Studentų 50, Kaunas, Lithuania

vytautas.stuikys@ktu.lt, renata.burbaite@ktu.lt, kristina.bespalova@ktu.lt, tomas.blazauskas@ktu.lt, dominykas.barisas@ktu.lt

Abstract: This paper introduces a Stage-Based (SB) Generative Learning Object (GLO) model to specify the learning content. Capabilities of the model are the content automatic generation and adaptation. Externally, our model has a similar structure as the known two-level generic models (i.e. metadata and content implementation). The internal structure, however, is quite different in both parts. The use of the external parameterization technology based on pre-programming predefines the internal structure. The SB model implements the deep internal staging by allocating parameters and functions (objects) into predefined stages according to the given context. The essence of the approach is the SB de-activation and activation of the objects within the specification. That ensures the automatic SB generation and flexibility for adaptation. We analyze the SB model capabilities, the use scenarios and processes, present a case study and extended results of using and evaluation in the robot-oriented computer science education.

Keywords: learning object, generative learning object, content generation and adaptation, stage-based model

1. Introduction

In technology enhanced learning (TEL), the educational content typically is called learning object (shortly LO or LOs). The term is known since 1994 due the W. Hodgins contribution. His intention was far-reaching – to resolve the problems related to systematization, interoperability and reuse of the educational resources. Now, after over two decades of its evolution, the LO-related research field has grown out into a separate branch in the TEL domain. With maturing of the field, a variety of LO definitions has been proposed. IEEE perhaps provides the most general one: "LO is any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning" (I. L. T. S. Committee, 2002). For other definitions, see (McGreal, 2004; Rossano et al., 2005; Wiley, 2000; Sosteric and Hesemeier, 2004).

Among the multiple LO types, the generative LO (shortly GLO, or GLOs) represents the innovative approach in TEL. The concept of GLO is due to the contribution of Boyle,

Morales et al. (Boyle et al., 2004; Morales et al., 2005), characterizing GLOs as "the next generation learning objects". The Center for Excellence in the design, development and use of LOs in UK (shortly, RLO-CETL) defines GLO as "an articulated and executable learning design that produces a class of learning objects" (Boyle et al., 2008). Therefore, this concept means a move from the component-based reuse model to the generative reuse model. The articulation in (Boyle et al., 2008) is conceived as (1) human-understandable explicit or implicit decisions involved in design for learning and (2) these decisions can be executed by computer software to produce LOs based on the design. In practice, i.e. when the GLO Authoring tool GLO Maker (http://glomaker.software.informer.com/3.0/) is used, the pedagogical designs are represented explicitly as the 'plug-in' patterns. The tool is used to create specific LOs based on the chosen pattern. Each of these LOs created in this way can be re-purposed by the local users, with the help of the same tool, to adapt the resources to their needs and preferences. Then all the LOs so created (or adapted) run as stand-alone Web based LOs. This approach has been borrowed from the systemic grammar (Boyle and Ravenscroft, 2012); however, it can be also viewed as the template-based approach in terms of software generative reuse (Sametinger, 1997).

The other generative technologies (such as programming languages and compilers, meta-programming-based approaches), in essence, are more powerful in their capabilities of automation. Therefore, they fit well to implement GLOs. The GLOs presented in (Štuikys and Damaševičius, 2007) are implemented using heterogeneous meta-programming (He MPG). In general, He MPG can be thought of as both a high-level programming technique (in terms of developing specifications executable by a computer), or as a generative technology (in terms of the tools used to support automation). The external parameterization applied on the internal content stands for the base principle to understand the essence of He MPG.

In more specific terms, at least two languages are used in the He MPG paradigm: meta-language and target language (Štuikys and Damaševičius, 2013). The first brings the notation for the external parameterization while the second represents the internal content to be manipulated by the constructs of the meta-language. The manipulation is automatic and parameterization therefore opens the way for a variety of possible modifications and adaptations. Typically, the internal content is a target program. It is also seen as a content for teaching, e.g. in Computer Science (CS)-related courses.

The aim of this paper is to focus on a specific MPG-based model, called stage-based GLO model (further SB GLO) that enables to implement indeed a deep and flexible pre-programmed adaptation, meaning automated adaptation of the content through stage-based generation and transformation. Those capabilities of the proposed model are the main contribution of the paper. Note that the SB GLO is derived from the initial MPG-based GLO. The latter is designed for reuse while the first is designed for reuse and adaptation.

The structure of the paper contains the following sections. In Section 2, we analyze the related work. In Section 3, we motivated our research methodology. In Section 4, we outline a framework to present overall aspects of the methodology without details. In Section 5, we present a background of the approach through the definition of basic terms and statement of assumptions and properties. In Section 6, we describe the context-aware stage-based GLO model in detail. In Section 7, we analyze a case study that includes the robot-based implementation of the model in solving the educational tasks to teach CS. In Section 8, we analyze the stage-based adaptation processes and learning scenarios. In

Section 9, we discuss capabilities of the methodology and present an overall evaluation with the focus on pedagogical aspects. Finally, in Section 10, we conclude the main results.

2. Related Work

We categorize the related work into two groups: 1) Context-related issues in TEL, 2) Approaches related to the model-driven stage-based development.

2.1 Context-related issues in TEL

As there is no common understanding of the term context, multiple definitions and views have been proposed so far. Among those, Dey (Dey, 2001) defines context as "*any information that can be used to characterize the situation of an entity*". By an entity, it is meant *"a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*." The paper (Verbert et al., 2012) gives an extensive analysis of definitions in relation to engineering of recommendation systems to support TEL.

Dourish indicates that context has a technical and social origin (Dourish, 2004). He argues that, from the social perspective, context is not something that describes a setting or situation, but rather a *feature of interaction*. Researchers in TEL say that this user-centered emphasis on factors affecting an activity is precisely what makes this notion of context meaningful for learning. From a technical perspective, context is understood as an *operational term* (Winograd, 2001). In this regard, papers (Schilit et al., 1994; Dey et al., 2001) define context by enumerating its categories as follows: *computing context* (such as network connectivity, etc.), *user context* (such as the user's profile, etc.), *physical context* (such as noise level, etc.), *time-related context* and *task-related context* (Verbert et al., 2012). The Zimmermann et al. (Zimmermann et al., 2007) operational view includes the following context categories: *individuality, activity, location, time, and relations*. Individuality is subdivided into four elements: *natural entity, human entity, artificial entity, and group entity*. This definition is perhaps one of the most comprehensive context definitions to date.

In TEL, such enumerations have also been proposed as an attempt to define the learner or teacher's context as an operational term. Many enumerations are defined for mobile learning. For example, Berri et al. (Berri et al., 2006) distinguish between technical and learner context elements. The first category deals with the technical aspects of mobile devices, their operational environment and constraints. The second category defines the learner context elements (e.g. aims and objectives of the learner, prerequisites, etc.). It is also essential to capture *interactions* between the *environment*, the *user*, their *tasks*, and other *users*. The paper (Azouaou and Desmoulins, 2006) aims at defining a context ontology of teacher's personal annotation, in order to use it in a context-aware annotation tool "MemoNote". The paper defines the active and passive contexts in the tool (annotation ontologies selection, annotation memorization, pattern definition and selection) to develop the complete teacher's context annotation ontology using the classical method specified for Protégé.

The *content granularity* and *context information* are related. Both are important factors to the efficiency and reusability of learning objects (LOs). The context information, e.g., is necessary to facilitate the discovery and reuse of LOs stored in global repositories or local libraries. Typically, LOs are incorporated into repositories without the context information. Users have to do some extension of the LO descriptions to fit their special use. Therefore, the paper (Man and Jin, 2010) introduces a context-rich paradigm, the related service driven tagging strategy and a context model of LOs. This model realizes the adaptive granularity of the content and LOs.

The paper (Huddlestone and Pike, 2005) describes a four-tier reusability model for making reuse happen in practice within organizations. The items that affect the viability of object reuse are the properties of the object itself (e.g. structural reuse and contextual reuse) and the organization's preparedness to undertake LO reuse (operational reuse and strategic reuse). *Structural reusability* is thought of as a function of how the object has been engineered. By c*ontextual reusability,* it is meant the applicability of the object to new learning events that affect on the potential audience size. *Operational reusability* has dependencies on organizational culture, personnel, procedures and technology. *Strategic reusability* is seen as a function of organizational strategy for systematic or opportunistic reuse.

2.2 The term stage and relevant methodologies

In the literature (typically, in SWE and CS fields explicitly and among e-learning implicitly), the term stage is used in two roles: (1) *as a time dimension* to split some process into parts and (2) *as a design principle and design process* itself. The process duration, however, may vary within wide boundaries. For example, the paper (Bastable and Dart, 2008) identifies the physical, cognitive and psychosocial characteristics of learners that influence learning at various *stages of growth and development* and also discusses appropriate teaching strategies. Also in e-learning, the term is used under the other name such as *level*. Indeed Bloom's taxonomy (Anderson et al., 2001), e.g., identifies the cognitive levels (they can be interpreted as stages) in knowledge gaining by learners, though the time dimension is implicit. A similar example is the methodology (Urquiza-Fuentes and Velázquez-Iturbide, 2009) to assess the knowledge of the learning process through engagement levels that include the following levels (stages): *viewing, responding, changing*, *constructing*, *presenting*. In terms of role 1, the paper (Rajlich and Bennett, 2000) deals with the *stage-based* software life cycle model that includes initial development, evolution, servicing, phaseout and closedown.

In the role 2, *stage and staging* can be thought of as the *separation of concepts* (also the term concerns is used), the well-known design principle since 1970 (Dijkstra, 1970), in which Dijkstra has applied information hiding and separation techniques to describe structural programming. Note also that Greer, for example, considers separation of concepts 'as a principle and a process' used in designing systems (Greer, 2008). We speak about that not only to explain the origin of introduced terms. We aim at highlighting the importance of the terms in TEL in general (in fact, the course designers and teachers use the terms, perhaps without the explicit naming). Therefore the explicit use of the terms (stage and staging) can found in the following contexts: stage programming in Taha works (Taha, 1999; Taha, 2004), stage-based meta-programming

(Štuikys and Damaševičius, 2013) and feature-based modeling. For example, the paper (Czarnecki et al., 2005) discusses *multi-stage configuration* of feature diagrams and the paper (Classen et al., 2009) proposes the *multi-level staged configuration* of feature diagrams to facilitate configuration in SW product line engineering (PLE). The other paper (Krueger, 2013) considers a *multi-stage configuration tree* proposed in the context of feature-based modeling for the 2nd Generation PLE. The latter supports the engineering, deployment and maintenance of product family trees. Feature selections and down selections are incrementally staged throughout the nodes in a product family tree.

2.3. Adaptation in e-learning

The paper (Brinton et al., 2015) discusses the design, implementation, and preliminary evaluation of Adaptive Educational System for personalized course delivery, which integrates lecture videos, text, assessments, and social learning into a mobile application. The system collects clickstream-level behavioral measurements about each student as they interact with the material. These measurements can subsequently be used to update the student's user model, which can in turn be used to determine the content adaptation.

The paper (Arai and Tolle, 2015) proposes a module based content adaptation approach for adapting composite e-learning web pages composed by *Microsoft (MS) Producer* tools for delivering the contents onto mobile learners.

The paper (Gutiérrez et al., 2016) presents: (i) a Sharable Auto-Adaptive Learning Object (SALO) that includes learning content and describes its own behaviour supported by dynamic languages; (ii) an example implementation of SALO for the delivery and assessment of a web development course using Moodle rubrics. As a result, the SALO can dynamically adapt their characteristics and behaviour in e-learning platforms.

The paper (Premlatha and Geetha, 2015) explores (i) the adaptation that can be based on learner context parameters, on the learning content (learning object) and the configuration of e-learning environment; (ii) provides a detail review about the various levels of adaptation, learning object design and process for learning content design, learner context parameters, and models/ components of e-learning; and (iii) analyzes the associations among the components, necessary to achieve the well-defined adaptation in e-learning environment.

The approach (Dorca et al., 2016) uses an expert system to implement a set of rules, which classifies LO according to their teaching style, and then automatically filters LO according to students' learning styles.

3. Motivation of the approach

The reuse concepts, first being borrowed from more matured domains (such as HW and CS) and then adapted to e-learning, dominate in TEL now. Similarly to other domains, TEL seeks for more effective solutions, especially in terms of seamless integration of pedagogical and technological approaches. In heterogeneous domains (TEL is just the case), context plays an extremely important role. There is a variety of contextual forms: the content-related, the technology-related, the social-related, and the pedagogy-related ones (Verbert et al., 2012). Content adaptation is a specific form of reuse always dependent on the context. The adaptation process is highly related on how it is represented and delivered. The general methodological principle is as such: before

delivering the whole content should be structured into parts to achieve the adequate granularity level and well-organized by sequencing (typical example is slides showing). In delivering the content (for audience, e.g. for students), the general pedagogical scenario can be outlined as follows: (1) Defining objectives explicitly; (2) Partitioning the whole content into parts; (3) Starting explaining simpler parts first and then moving to more complex items; (4) Choosing items for delivery (from possible variants, either simple or complex) so that the selected variant would be more relevant to a particular interest (it may be treated as context) to the audience.

What does the presented scenario in essence mean? In fact, the first motivates the whole activity. The second means the physical staging, though the other terms can be used such splitting, decomposing, partitioning, etc., but we prefer to use *staging* here. The third means both the staging and sequencing and as well as an intent for adaptation. Finally, the fourth means a real action to support adaptation. Of course, to be viable, this scenario should be well-planned in advance. Therefore, the scenario describes the relationship chain: *staging-sequencing-adapting*. However, there is the fourth item, *context*, which is influential to the whole chain. The context may predefine the way on how the chain should be formed and used. As motivation and staging appear at the beginning, we can call the chain as *pedagogical staging* (for more detail, see Case Study in Section 7). In fact, the pedagogical staging motivates the need of using technological staging if one wants to achieve aims of automated adaptation. One can learn more on technological staging from stage-based (SB) programming (Taha, 1999; Taha, 2004) and SB meta-programming (Štuikys and Damaševičius, 2013).

Here, by technological staging, we mean the technological capabilities first to specify the content in *stages explicitly* and then to interpret the specification using the adequate tool. The tool brings automation. As we focus on automated adaptation, the following question arises. Where and how should meet each other two concepts: pedagogical staging and technological staging? We argue here that the meeting point should be *a stage-based model*. The latter is derived from the initial MPG-based GLO.

Externally, both models have a similar structure as known two-level generic models (i.e. *metadata* and *content implementation*). The internal structure, however, is quite different in both parts. The use of the external parameterization technology based on pre-programming predefines the internal structure. Furthermore, the structure is derived from the initial parameterized GLO model using the refactoring tool. Typically the generic LO model has two levels: *meta-level* and *implementation level*. The meta-level serves for *delivering data to* the implementation level to support such processes as *search* and *generation.* The implementation level serves for the physical realization of the processes with the help of any computing environment (or technology). Therefore, the generic model has two parts: *metadata* plus *implementation* of the functionality.

The SB model GLO differs from the initial one by the following attributes: (i) internal structure is multi-staged and derivative from the initial one; (ii) the content generation is the multi-staged process generating other GLOs having the context-related functionality until the LO is created; (iii) the generation process is governed by the context model semi-automatically or even automatically.

Why the SB model is needed? As the initial model GLO is designed for reuse with the extended reuse extent through enlarging the variability space, the adaptation of the model to the concrete context of use lacks of flexibility and requires intensive manual

efforts. The SB GLO has two essential advantages: (1) separating the different context information (e.g. teacher's, student's, technological) through staging explicitly and (2) solving the adaptation problem semi-automatically or even automatically. Note that the initial short version of the approach can be find in (Stuikys et al., 2016).

4. A framework and tasks

The approach and tasks we discuss in this paper are indeed complex in their own rank. The complexity is due to the multiple reasons such as: (i) heterogeneity of the TEL domain itself, (ii) a high coupling of both the pedagogical-social and technological issues, (iii) the complexity of the context information to be considered in the process, and (iv) technological issues to tackle tasks through automation, to name a few.

We therefore introduce a framework to better understand our approach. We define the framework in a reuse-oriented manner. In essence, reuse among other issues is also concerned with the time dimension, though implicitly (say as a context). As a result, we consider the framework as a life cycle of our objects (GLOs). The life cycle includes four main processes (see Fig. 1): *design, refactoring, generation* and *use* (*learning*). Each process contains within the adequate sub-processes. For example, the design covers the TEL domain modeling and the development of GLO specifications using the model transformation approach (see (Štuikys, 2015), for details). We treat the process as *design-for-reuse* (also meaning the potential for a wide-scale adaptation). The process results in creating the *local GLO library* to cover topics of the whole course (e.g. CS in our case). We consider the next process (refactoring) as *design-for-adaptation*. We treat the generation process as *design-with-adaptation*. The latter in fact fuels the learning process. The framework also outlines actors (their responsibilities and actions) and tools used within the life-cycle model.

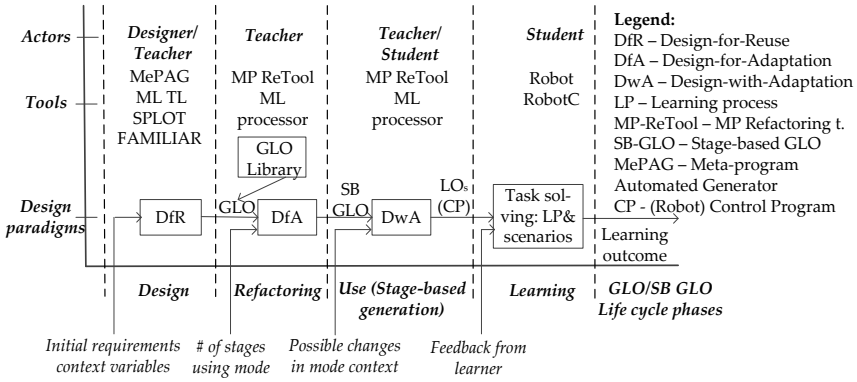


Fig. 1. A framework of the approach: GLO and SB GLO life cycle.

The model (i.e. SB) we discuss in this paper falls within both processes (refactoring and learning). More specifically, refactoring is the process to transform the initial GLO (obtained as a product of the design process) into the representation of SB model. The *use* is about the teaching and learning process using the initial GLOs, SB GLOs and adapted LOs. The latter is an actual content to be learned by students.

Now we are able to extend the framework and formulate the tasks to be considered. First, we need to define the reuse extent. To do so, we focus on the TEL domain itself, because the reuse potential is predefined by the artefacts relevant to the domain. The TEL domain is highly heterogeneous and includes pedagogical, social, technological and content aspects, each containing multiple variants (Štuikys, 2015). Therefore, the variability space is indeed huge and expanding continuously. To which extent we are able to recognize, to extract and to represent the knowledge explicitly within this variability space, in the same or similar extent we can automate the domain. This statement is the base of generative reuse (Frakes and Kang, 2005). It was well-known in software reuse for a long time (e.g., since (Prieto-Diaz, 1988) and now (to some extent) in e-learning (Polsani, 2006; Boyle, 2002).

The explicit learning variability model is the only one side of the problem. We need to have yet another component - the relevant generative technology (He MPG in our case) to implement the learning variability model. In general, the learning domain automation is the common task we consider in this paper. More specifically, the task covers the content generation and automated adaptation. As both are highly dependent in our approach, we formulate the content generation-adaptation task as the mapping of learning variability model onto the generative technology model.

In fact, the implemented variability predefines the reuse scope and extent for adaptation. The pre-programmed GLO model, discussed at the very abstract level in Section 1, has been just implemented on this conceptual basis. Parameters and their relationships are objects to express the variability aspects. If we take into account the implicit context (e.g. pedagogy related parameters have a higher priority with respect to the others), the model itself has the potential for adaptation. However, to make the adaptation more flexible, we need to introduce (1) the explicit context and (2) to re-arrange the model structure by introducing the stage-based model. The explicit context is the prerequisite to manage the stage-based generation and automated adaptation.

We consider the task not from scratch, but from the point at which we already have a pre-programmed GLO specification as an input data for implementing automated adaptation. The design of the specification is not the topic of this paper (the design methodology can be found in (Štuikys, 2015). We also assume that the initial specification is syntactically and semantically correct and includes the explicit context too. Furthermore, the specification implements a high degree of variability space, meaning its potential use by teachers and students within the same course.

5. A background of the approach

The background covers the assumptions, definitions of basic terms and properties that are concerned with both the initial GLO and SB GLO.

5.1 Basic Assumptions

Assumption 1. Explicit knowledge on educational aspects in large (meaning to support a high reuse extent) should be extracted through *analysis and variability modeling* (both are reuse-driven activities) to be performed in advance by the domain expert.

Assumption 2. The initial GLO model is correct and the correctness is approved by modeling and experimental validation.

Assumption 3. The interface of the initial GLO (it can be thought also as metadata) additionally supplies comments on the context selection.

Assumption 4. The number of eligible stages should be obtained by the tool and the number of needed stages – by the user (typically by teacher).

5.2 Definition of basic terms

First, we define the terms related to the parameterized GLO and then the stage-based GLO definitions follow.

Definition 1. Parameter is the unified representation of some educational aspect, such as pedagogical, social, technological, or content (see *Assumption 1* and *Property 1*).

Definition 2. The parameter's value is the concrete value of the parameter (see *Property 2*).

Definition 3. Parameter's context is the value expressed through a *fuzzy variable* explicitly and allocated to the parameter. Fuzzy variable is a value taken either from the *short set of priorities* {HP, IP, LP} (where HP - High Priority, IP - Intermediate Priority, LP – Low Priority), or from the extended set of priorities (see also *Property 3 and Property 4*).

Definition 4. The interface of parameterized GLO is a set of the contextualized parameters along with their values (see also *Assumption 3*, *Property 5 and 6*).

Definition 5. The function is the (*meta*-)*language construct* to define a possible manipulation (insertion, deletion, change, etc.) through the parameter-function relationship (see also *Property 7*).

Definition 6. Implementation of the GLO specification is the set of the predefined parameter-function relationships.

Definition 7. The initial GLO model is the specification containing two parts: interface (*Definition 4*) and GLO implementation (*see, Definition 6*).

Definition 8. Generation is the process of following actions: (a) a manual selecting of parameter values for each parameter, (b) executing the specification by the tool (i.e. processor of the meta-language in which the functions were described).

Definition 9. Adaptation is the process of following actions: (a) context-related selection of parameter values for each parameter by the user (teacher or student), (b) automatic executing the specification by the tool (i.e. processor of the language in which the functions were described) (see *Assumption 2*).

Definition 10. Stage is an abstraction to specify the *context dependent part* of the whole process (e.g. generation or adaptation). The whole process is defined as multi-stage one.

Definition 11. Staging is the process in which the initial GLO model is expressed and specified by stages using the adequate support (the relevant technology, such as the language functions, and the adequate mechanism within the language processor, such the function de-activation).

Definition 12. Stage-based (aka multi-stage) model is the model describing the way on how parameters and functions are allocated to stages without intersection (see Fig. 3).

Definition 13. Construct (i.e. parameter or function) is either in the *active* or *passive*

state. The construct is said to be *active* if it, when executed by the processor, performs the prescribed action (have no symbol "\" to deny the state, e.g. *f*(*p*)). The construct is said to be *passive* if it cannot perform the prescribed action due to the denying symbol"\" written before the construct (e.g. *f*(*p*) the function *f* and its parameter *p* are passive) (see also *Property 9*).

Definition 14. Stage-based generation is performed by the language processor. It is the process of evaluating parameters at the top stage (say *k* stage) first. The evaluation results in: (a) the change of the specification according to the values of the selected parameters; (b) the decrease of the number of stages by 1 (meaning also the decrease of passiveness at the remaining stages); (c) the intermediate (narrowed) specification. Next, the process is repeated until the stage 1 is achieved (see also *Properties 10-11*).

Definition 15. Stage-based adaptation is the process of selecting the values of the context-driven parameters by the user at each stage gradually and then invocating the stage-based generation process.

Definition 16. Stage-based GLO specification is the derivative specification created using the adequate refactoring tool that transforms the initial specification into the stage–based one according to the model (Fig. 3) and using the staging strategy defined by *Definitions 10-13* (see also *Assumption 4, Properties 10-11*).

5.3 Basic properties

Property 1. Parameters are represented uniformly, but they differ in semantics, the latter being recognized from the context.

Property 2. Values of different parameter may interact (i.e. to be dependent). If that is the case, the interaction is expressed through constraints *requires* (e.g. **beginner** *requires* **simple content**) and *excludes* (e.g. **topic 1** *excludes* **topic 2**, i.e. can be used only one at a time). Otherwise, these parameters are not interacting, i.e. are independent.

Property 3. Typically, pedagogy-related parameters have the high priority (HP) and the content-related parameters have the low priority (LP). However, there might be more complex the parameter-context relationships (Štuikys, 2015), not considered here.

Property 4. Context can be pre-programmed and expressed through a set of fuzzy variables (such as HP/IP or IP/LP) to form the parameter-context relationship.

Property 5. As some parameters may interact, we need to consider the parameter groups. Therefore, the interaction may appear within a group, but not among groups. A group may also consist of a single independent parameter. The context information is defined not for a separate parameter (if it is a member of the group), but for the whole group.

Property 6. In general, the interface of GLO is a set of context-aware parameter groups. The interface predefines the *variability space* for possible adaptation.

Property 7. The argument of a function to implement the GLO functionality may be the parameter, the fragment of the content, the other function, or a combination thereof.

Property 8. There are constraints to perform staging such as the one: the interacting parameters should appear at the same stage.

Property 9. The language processor performs the action of changing the construct state from passive to active by deleting the symbol "\" through the one pass of executing the specification. However, if the construct has multiple symbols "\" at some

intermediate stage, it remains passive, but the "degree of its passiveness" is decreased by 1.

Property 10. If stages are numbered as (k, k-1, 1), where k is the number of the top stage, all constructs at the stage k are active, while the remaining are passive with the growing degree of "passiveness" at each subsequent stage.

Property 11. Staging decreases *readability of the* stage-based specification due to the use of multiple symbols "\" to de-activate the constructs. However, there exists the exact relationship between the stage number and the "degree of passiveness" at the given stage, expressed through the de-activating symbols (Štuikys, 2015).

Because of this and other stated properties, it was possible to build the tool providing the automatic generation and adaptation. The tool (Bespalova et al., 2013) fully hides the technological mechanism of staging (for more details, see Case Study, Section 7).

Definitions from 1 to 16 are concerned with the initial GLO model and its technological staging. *Assumption 1* and *Property 1* are concerned with the integration of pedagogical and technological staging. Fig. 2 illustrates the initial GLO model in more detail. The graphical symbols (see Legend) explain the terms before defined formally. Fig. 3 illustrates the stage-based model. It is a derivative model derived from the initial based on the presented background. Again, the adequate formally defined terms are explained there by graphical symbols for better understanding (see Legend, in Fig. 3). Note that shading is an abstract representation of the stage's de-activation process.

Basically, we apply both models in CS education using robots. From the given description, one might receive the impression that the approach is relevant to CS education only, though we try to use not so much specific CS terms, but rather the general terms, such as teaching content. The basic teaching content in CS is computer (robot) programs treated as *textual* LOs. However, they also may contain pictures, diagrams or movies within (those are to be either modifiable or non-modifiable). Therefore, the models might be applied to other kind of LOs if the following condition holds: the explicit variability model is known for those contents.

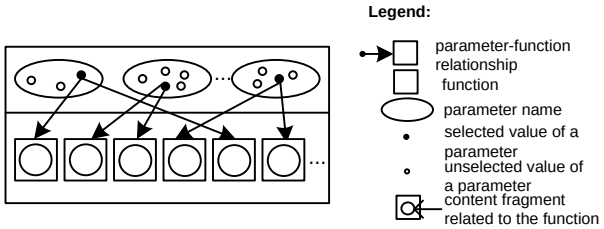


Fig. 2. Initial parameterized GLO model.

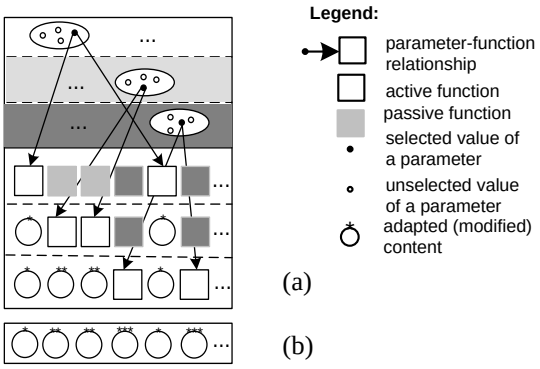


Fig. 3. Stage-based GLO model (a) and the LO instance derived from the model (b).

6. Staging and context-awareness

In general, learning attributes, i.e. parameters in terms of GLOs, fall into four categories (adapted from (Koehler and Mishra, 2009)): P-Pedagogy-related (i.e. teaching goal, teaching model, etc.), S-Social-oriented (e.g. student previous knowledge, abilities, etc.), T- technology-oriented (e.g. characteristics of educational robots when teaching is based on this technology), C- Content-related (e.g. algorithms to realize the CS teaching tasks using robots). As those categories differ in semantics and roles, an order (i.e. priority) of their interpretation is to be introduced, when LOs are designed and used. We can express their ordering by the relation:

$$P \prec= S \prec T \prec= C, \tag{1}$$

Here, the record $X \prec= Y$ means that X has the same or higher priority with respect to Y. (The relation $X \prec Y$ means a strong priority). In fact, this relation can be treated as context information. It is more convenient, however, to represent the context more directly through priorities taken from the set of fuzzy variables:

$$W = \{HP, IP, LP\} \text{ or } W = \{HP \prec IP \prec LP\} \tag{2}$$

$$W = \{HP, IP_1, IP_2, LP\} \text{ or } W = \{HP \prec IP_1 \prec IP_2 \prec LP\} \tag{3}$$

Here, fuzzy variables have the following meaning: HP -High Priority, IP1 – Intermediate Priority first (higher), IP2 –Intermediate priority second (lower), LP –Low Priority. The set W being defined through fuzzy variables is treated as context. As some categories of parameters are indeed closely related or, in some other cases, they can be treated as the ones (e.g. teacher-based and student-based are indeed pedagogical, or technology and content), we are able to consider typical cases in describing the parameter category – context relationships (see Fig. 4, a-e are possible variants). But there is the other sort of relationship, i.e. parameter-parameter relationship (among different categories and/or among parameters of the same category). We call any kind of parameter relationship as parameter interaction or dependency (see Property 2). It is convenient to represent the parameter dependency groups by tree-like graphs, where nodes represent parameters and branches represent interactions among the parameters. In Fig. 5, we present all possible variants of the parameter interactions abstractly, ignoring

the categories of parameters. The variant *a* represents not interacting parameters (*g*=5 defines the number of independent groups). The variant *d* represents a theoretically possible case when all parameters are interacting. However, the most practical variants are *b* and *c*, where 4 and 3 not interacting parameter groups are given respectively. The number *g* is important, because it also specifies the number of eligible stages. It is so because the dependent parameters should appear at the same stage when interpreted (otherwise the interpretation would be erroneous).

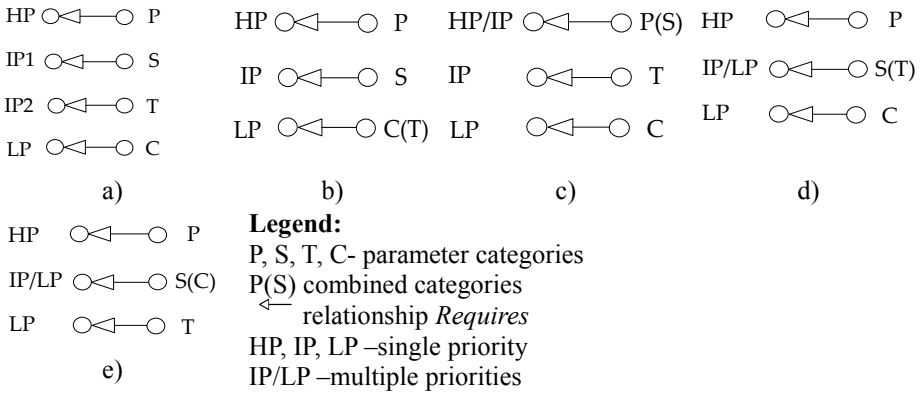


Fig. 4. Relationship between context (priorities) and parameter categories.

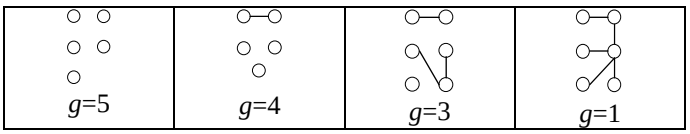


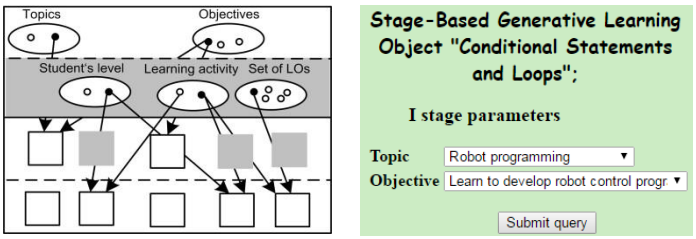
Fig. 5. Relationship among parameter groups (g- # of groups).

7. Case study and results

The aim of the case study is to demonstrate the viability of the implementation of the SB GLO model in the real robot-based educational setting to provide the course "Programming Basics" for the 10th grade secondary school students. The preconditions for that are: (a) the local library with the initial GLOs already exists and it covers (partially or fully) the whole topics of the course; (b) the teacher first selects the initial GLO specification from the library, identifies the needed number of stages and using the adequate tool transforms the initial specification into the SB GLO. However, SB GLOs can be prepared in advanced and taken from the library (if the existing item fits the educational needs, e.g. the number of stages; otherwise the teacher creates the SB GLO anew using the refactoring tool (Bespalova et al., 2013)). Note that the tool may be used in two modes: (a) automatic staging (the stages are formed using the context information {HP, IP, LP}); (b) manual staging (user is able to allocate parameters to stages

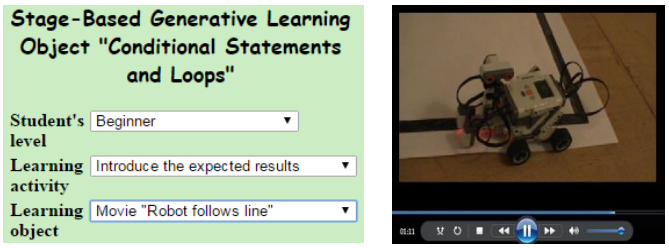
interactively according to his/her vision).

Fig. 6 (a) outlines the abstract implementation vision of the GLO model for the task "*Following the line with obstacles*" by robot to learn the topic "*Conditional statements and loops*". For simplicity, we consider the two-stage GLO here (the context information is not shown). Fig 6 (b) shows the top stage (I) user interface after selecting the adequate parameter values. Fig. 7 (a) shows the stage II parameters of the initial GLO and makes the selection from the menu. The execution of this GLO generates the result (Fig. 7 (b)) adapted to the *Beginner,* i.e. the movie as an *expected result* that will be really achieved after learning (creating and executing the robot's control program through *Practice*). Therefore, the movie is LO for the *Beginner*, being generated through the two-stage process.



(a) (b)

Fig. 6. Two-stage GLO model: (a) abstract implementation vision; (b) vision through the user interface (stage II is hidden).



(a) (b)

Fig. 7. Abstract vision after evaluation of the top stage (according to values of Fig. 6 (b)) (a) user interface to evaluate the stage II; (b) movie, illustrating the expected result to motivate the learning task "Following line with obstacle" by robot

Now let us go through the other adaptation path by selecting the parameter values. Assume that the student made the selection at the execution stage II: "Intermediate", "Practice", "GLO Following the line with obstacles" (see Fig 8 (a)). Then the system opens the interface with the multiple menus (see Fig. 8 (b)). Then student inserts own values according to the task as it is shown in Fig. 8 (b). The result of processing is the control program (CP) with conditional statements and loops in RobotC (its fragment is given in Fig. 8 (c)). Then the generated CP is to be loaded into the robot's memory and the student is able to monitor the execution of the CP and get not the "motivating movie", but the line following produced by the robot in real time. And this result might be quite different from the "motivating movie" depending on the task.

The case study represents the only one SB GLO taken from the created local library. A full list of the initial GLOs within the library includes 16 representative objects that cover "Programming Basics" course and other CS-related courses for the secondary school students.

The created GLOs can also be useful, to some extent, in teaching other subjects, such as mathematics, physics, and engineering. For example, the task "Ornament drawing" is based on mathematical calculations of the trigonometric functions, the time-velocity dependency predefines the robot's movement (knowledge in physics), and students can also be involved in constructing the educational robots to gain the engineering skills.

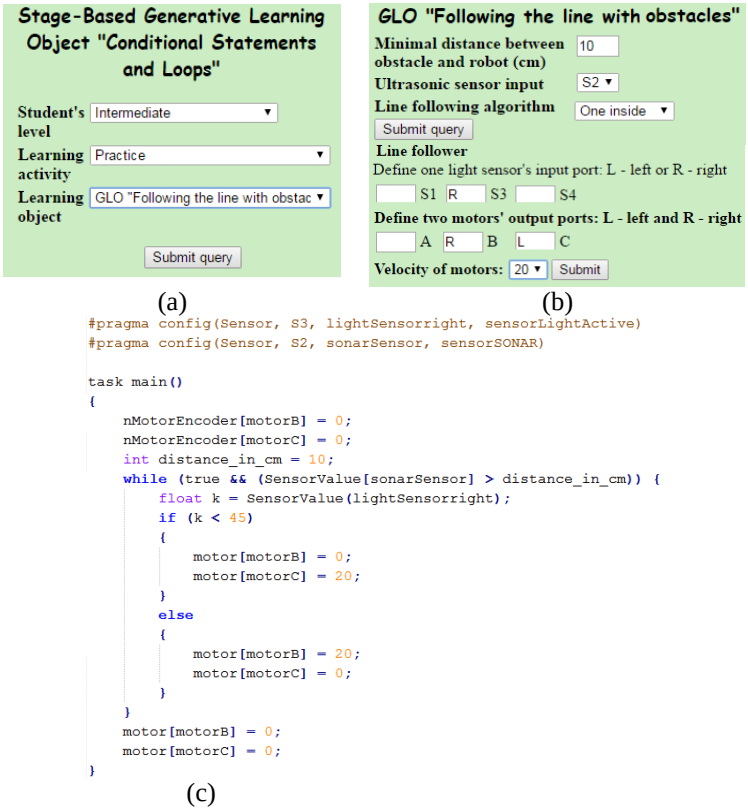


Fig. 8. Selection made at the execution of stage II (a); interface and menu induced by the value "Following the line with obstacles" (b); a fragment of the robot control program (c).

8. Stage-based adaptation processes and scenarios

Before considering those issues, firstly we define terms *surface learning*, *deep learning* and *active learning*. According to Houghton (Houghton, 2004), surface learning is "*accepting new facts and ideas uncritically and attempting to store them as isolated, unconnected items*". And deep learning is "*examining new facts and ideas critically, and*

tying them into existing cognitive structures and making numerous links between ideas". According to (Yang, 2013), active learning is "*a process whereby students engage in activities, such as reading, writing, discussion, or problem solving that promote analysis, synthesis, and evaluation of class content*". Note that educational robots promote active learning because there is the possibility to combining active learning methods.

Now we are able to present the stage-based adaptation process in learning in more detail. In Fig. 9, we outline the approach schematically as a multiple process with different sorts of adaptation scenarios and feedbacks. The top part relates to the teacher's context, while the other – to the student's context. Here, we show the stage-based specification abstractly through stage numbers (top stage *k* is for the teacher). There are three kinds of adaptation scenarios: i) stage-based at the *surface learning phase*, ii) *technological* (i.e. intermediate) *phase* and iii) *adaptation at the deep learning phase.*

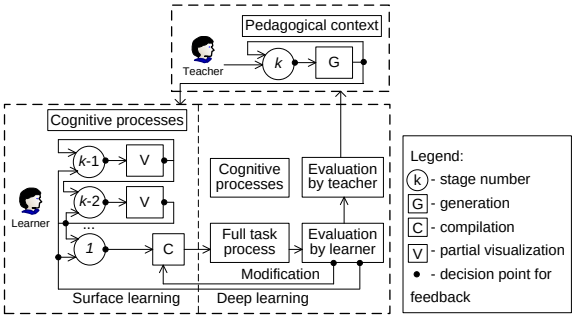


Fig. 9. A scenarios of LO adaptation and cognitive-based processes.

What is the meaning of those scenarios? The surface learning and cognition process starts, when the learner gradually moves through stages (from *k*-1 to 1). At each stage, the learner introduces the parameter values via menu with a self-reflection on those values. Then the generation process with a possible partial visualization follows to represent the result of the *partial compilation*. The latter is possible because, after activating parameters at the adequate stage, a corresponding fragment of the target program (RobotC in our case) is created. And the learner can see the fragment visually along with a remaining meta-code yet not being instantiated. The *complete compilation* (C in Fig. 9) is possible only after the full instance is generated in stage 1.

How the processes in surface learning are supported? The meta-language processor is the tool to support the *stage-based generation* and *partial compilation* of the adapted GLOs. The complete compilation differs from the partial compilation by the tool used and the product produced. The RobotC compiler performs the complete compilation, meaning the creation of an executable program (robot CP) to provide the learning task in the real setting.

Where is the technological scenario? When the learner uses the RobotC compiler to do the complete compilation, he/she *changes the* technological environment, because typically the learner uses PC for managing the SB GLO specifications through the Internet and GLO library. This scenario also includes loading the robot CP into its memory. Therefore, we are able to treat the technological scenario as a bridge to link the surface and deep learning.

What is about deep learning? This scenario starts, when the learner observes the robot's operations. The task solving with the help of robots in the real setting has many technical aspects (such as visible characteristics related to sensors, motors used, their velocities, etc.) and social attributes (such as a curiosity to know whether or not everything is going as was planned, etc.). Therefore, deep learning is concerned with self-reflecting on what is going on the screen, analyzing, discussing, formulating questions, trying to improve the robots actions. All these may require changes and repeating processes. The indicated multiple feedbacks serve for this role. Therefore, we define the active learning as the integration of introduced scenarios using the feedback links.

9. Analysis of capabilities of the SB model

9.1 Designer's perspective

The designer may focus on two aspects: (1) how the initial parameterized GLO and (2) how the SB GLO should be created. As it was already stated (see Section 5, definitions), the learning variability space is the main concept on which both models rely. In fact, the variability space is the *TEL domain model* to be created in advance through analysis and modeling. As the domain is highly heterogeneous, most likely the expert knowledge taken from different sub-domains are needed in creating the domain model. For example, the most crucial knowledge might be required in defining the *interaction* among the pedagogical, social and content attributes. The creation of the semantically correct domain model is the responsibility of the domain expert. The designer should be aware that the domain model is correct, or otherwise, should address to the domain expert for advice and corrections. The designer responsibility is to map the domain model onto the solution domain model in creating the initial GLO. That can be done manually or using the adequate tools (Bespalova et al., 2013).

From the designer's perspective, the parameterized GLO along with the language processor, in which the parameter-language relationship is coded, is the *generator* of the LO components on demand. The task of creating SB GLO becomes extremely error-prone, when the number of stages is more than two. Therefore, the adequate tool is needed (Bespalova et al., 2013).

From this viewpoint, SB GLO is the generator of the others narrower SB GLOs. The generation process is first implemented by the specific refactoring tool transforming the initial GLO into the stage-based one. Having the stage-based specification, the language processor stands for the meta-generator.

What is the experience of the designer (i.e. CS teacher) in using the model? At the beginning (in 2012) two-stage GLOs were created manually. Later (in 2014) the tool to design the SB GLOs was created (Bespalova et al., 2013; Burbaite et al., 2014).

9.2 Teacher's perspective

The teacher is a top-level user of both the initial and SB GLOs. How can the teacher interpret a single initial GLO? The specification can be seen as a full set of LO components to be generated from the predefined variability space by selecting all

defined parameter values. The generated components are related and differ only in some aspects predefined by parameter variability. If we have, for example, 5 independent parameters each having 4 variants, the variability space is equal to 4 * 4 * 4 * 4 * 4 = 1024, and the number of components is the same. The generated components as a learning resource can be located in a local library supplied by metadata for search. Therefore, it is possible to make juxtaposition between the library and the parameterized (pre-programmed) GLO specification. The following pairs of items (the first belongs to the library, the second – to the GLO) are under the focus:

- Metadata – parameter;
- Search through metadata – generation through parameterization;
- Multiple explicit components – one specification with the multiple components specifically woven inside the specification;
- Typical case of the component-based reuse – typical case of generative reuse;
- The library scaling problem (Biggerstaff, 1994) may occur – no such a problem or its effect is highly reduced;
- Context is implicit (search errors may occur) – context is explicit, i.e. context-awareness is at hand (erroneous generation is excluded);
- Library maintenance requires a substantial handwork – maintenance is easier.

How can the teacher interpret *a set of the initial GLO* in terms of the teaching process? The set may cover the whole course, perhaps with some specific examples of LOs taken from the digital libraries. Therefore, the teacher is able to create his/her own local library, containing mixed LOs, generative and component-based. What is the role of SB GLOs? Using the refactoring tool (it transforms the initial GLO into SB GLO), it is possible to create *the generative local library* with the much more capabilities for *adaptation on demand.* And those capabilities can be expressed explicitly through the concrete context. For example, if the pedagogical context contains two-teaching activities (e.g. *case study*, *practice*), then two GLOs, separate for each context, are generated. If the teacher wants to adapt this specification further to a particular group of students satisfying their specific learning interests, the teacher may move to the next (lower) stage and repeat the generation/adaptation process. For that, the teacher needs first to select from the menu the *context-aware* parameter values relevant to those interests and then initiate the run of the language processor. There is no need of knowing the internal structure of the SB GLO by the teacher. Always the teacher (if he/she has no knowledge to act as co-designer of the specification) works with the SB GLO as the black-box entity.

What is the experience of teacher in using the model? We have started creating and using the parameterized GLOs in 2011 and context-aware SB GLO in 2014.

9.3 Student's perspective

Again, students work with the 'narrowed parameter space' of GLO already adapted to their context through staging. The working mode is similar as the teacher's: students see the graphical interface and perform the parameter selection relevant to their needs. The student is able to create his/her personal library of GLOs (or LOs) specifically oriented to his/her profile.

Table 1. Results of the students' evaluation

1. ***At which extent the methodology was useful? (only one answer possible)***

Student choices	GLO		SB GLO	
Very useful	28	36 %	17	45 %
Useful	32	40 %	21	55 %
More useful than non-useful	16	20 %	0	0 %
More non-useful than useful	4	6 %	0	0 %
Non-useful	0	0 %	0	0 %
Totally non-useful	0	0 %	0	0 %

2. ***What was the most interesting within the methodology? (multiple answers possible)***

Student choices	GLO		SB GLO	
Interesting tasks	48	31 %	24	28 %
New learning way	39	25 %	24	28 %
Learning is easier and faster	22	14 %	10	12 %
Fault-tolerance	16	10 %	10	12 %
Stimulate thinking	31	20 %	17	20 %

3. ***What knowledge and competence you were able to improve using GLOs? (multiple answers)***

Student choices	GLO		SB GLO	
Programming	58	38 %	31	34 %
Mathematics	27	18 %	24	26 %
Logic thinking and cognition	36	24 %	33	33 %
The practical evidence on how the task is solved	8	5 %	4	4 %

4. ***Where and when the use of GLOs should be targeted?(multiple answers possible)***

Student choices	GLO		SB GLO	
Always in each lesson on programming	30	27 %	10	19 %
In other courses (mathematics, physics)	21	19 %	14	27 %
Sometimes for lesson variation	51	46 %	24	47 %
For generalizing the topic	9	8 %	4	7 %

5. ***For what student's abilities the use of GLOs fits best? (multiple answers possible)***

Student choices	GLO		SB GLO	
Low abilities	12	8 %	11	13 %
Adequate abilities	46	31 %	21	25 %
High abilities	56	38 %	35	42 %
Very high abilities	34	23 %	17	20 %

We have created a questionnaire to assess the students' opinion on using GLOs (SB GLOs) in the learning process. The respondents (in total: 80 who used GLOs during 2014-2015 and 38 who used SB GLOs during 2015-2016) were secondary school students of the 10^{th} grade (15-16 years old). Results are given in Table I. Results are calculated taking into account the total number of answers (in the case of multiple answers). Note that there was no specific intent to make a distinction between the parameterized initial GLO and its derivatives (i.e. SB GLOs).

We can conclude that students treat the methodology as a useful means for their active learning, because the methodology supports to some extent the interdisciplinary aspects of GLOs. Some students, however, have achieved the knowledge level of deep learning only, though that was not measured explicitly. More on overall evaluation can be found in (Štuikys et al., 2016).

10. Conclusion

Typically, using the component-based reuse model, LO content is obtained through searching within digital libraries. The search procedure is automatic, but its result highly depends on the accuracy of query. The adaptation follows after the search, often due to the unsatisfactory search result. In this case, the adaptation is performed manually. The generative reuse model, when implemented as the parameterized GLO correctly, enables to substitute the search procedure (partially or fully) by the generating process. The latter always gives the accurate result taken from the predefined content-context variability space through the precise parameterization. By selecting the predefined context-aware parameter values, it is possible to perform the adaptation semi-automatically, or to some extent automatically. However, the derivative GLO model, called stage-based one, which was discussed in the paper, has the enhanced possibility for adaptation. By introducing staging, it is possible to separate the teacher and student context and technological context from the content’s context explicitly. Then the selected variant is generated automatically on demand.

The introduced approach also has some limitations and difficulties. It requires the precise domain-level model. Its creation requires a great deal of efforts and heterogeneous knowledge. However, those efforts can be loaded not on teacher or student, but on the domain expert and designer. Furthermore, this knowledge can be reused multiple times. The adequate technological support (domain modeling and model transformations), to exploit the capabilities of the approach, is not yet sufficient and perfect. Those issues are seen as a future work.

References

Anderson, L. W., Krathwohl, D. R., Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon.

Arai, K., Tolle, H. (2011). Module based content adaptation of composite e-learning content for delivering to mobile learners. *International Journal of Computer Theory and Engineering*, *3*(3), 382.

Azouaou, F., Desmoulins, C. (2006). Using and modeling context with ontology in e-learning: the case of teacher's personal annotation. In *Workshop on Applications of Semantic Web Technologies for e-Learning (SWEL@ AH'06), Dublin, Ireland*.

Bastable, S. B., Dart, M. A. (2008). Developmental stages of the learner. *Nurse as educator: Principles of teaching and learning practice*, 147-198.

Berri, J., Benlamri, R., Atif, Y. (2006, July). Ontology-based framework for context-aware mobile learning. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing* (pp. 1307-1310). ACM.

Bespalova, K., Burbaitė, R., Štuikys, V. (2013). MePAG tools. http://proin.ktu.lt/metaprogram/MePAG/

Biggerstaff, T. J. (1994, November). The library scaling problem and the limits of concrete component reuse. In *Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on* (pp. 102-109). IEEE.

Boyle, T. (2003). Design principles for authoring dynamic, reusable learning objects. *Australian Journal of Educational Technology*, *19*(1), 46-58.

Boyle, T., Ravenscroft, A. (2012). Context and deep learning design. *Computers & Education*, *59*(4), 1224-1233.

Boyle, T., Leeder, D., Chase, H. (2004, November). To boldly GLO–towards the next generation of learning objects. In *World Conference on eLearning in Corporate, Government, Healthcare and Higher Education, Washington USA, Nov*.

Boyle, T., Ljubojevic, D., Agombar, M., Baur, E. (2008, June). The conceptual structure of generative learning objects (GLOs). In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 4570-4579).

Brinton, C. G., Rill, R., Ha, S., Chiang, M., Smith, R., & Ju, W. (2015). Individualization for education at scale: Miic design and preliminary evaluation. *IEEE Transactions on Learning Technologies*, *8*(1), 136-148.

Burbaite, R., Bespalova, K., Damasevicius, R., Stuikys, V. (2014). Context Aware Generative Learning Objects for Teaching Computer Science. *International Journal of Engineering Education*, *30*(4), 929-936.

Burbaite, R., Bespalova, K., Drasute, V., Ziberkas, G., Venckauskas, A. (2016a, June). Stage-Based Generative Learning Object Model to Support Automatic Content Generation and Adaptation. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual* (Vol. 1, pp. 712-721). IEEE.

Classen, A., Hubaux, A., Heymans, P. (2009). A Formal Semantics for Multi-level Staged Configuration. *VaMoS*, *9*, 51-60.

Czarnecki, K., Helsen, S., Eisenecker, U. (2005). Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice*, *10*(2), 143-169.

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, *5*(1), 4-7.

Dey, A. K., Abowd, G. D., Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, *16*(2), 97-166.

Dijkstra, E. W. (1970). Notes on structured programming.

Dorca, F. A., Araujo, R. D., De Carvalho, V. C., Resende, D. T., Cattelan, R. G. (2016). An Automatic and Dynamic Approach for Personalized Recommendation of Learning Objects

Considering Students Learning Styles: An Experimental Analysis. *Informatics in Education-An International Journal*, (Vol15_1), 45-62.

Dourish, P. (2004). What we talk about when we talk about context. *Personal and ubiquitous computing*, *8*(1), 19-30.

Frakes, W., Kang, K. (2005). Software reuse research: Status and future.

GLO Maker. Authoring tool for learning objects. Available: http://glomaker.software.informer.com/3.0/

Greer, D. (2008). The art of separation of concerns. *Web log post*.

Gutiérrez, I., Álvarez, V., Paule, M., Pérez-Pérez, J. R., de Freitas, S. (2016). Adaptation in E-Learning Content Specifications with Dynamic Sharable Objects. *Systems*, *4*(2), 24.

Houghton, W. (2004). *Engineering subject centre guide: Learning and teaching theory for engineering academics*. © Higher Education Academy Engineering Subject Centre, Loughborough University.

Huddlestone, J., Pike, J. (2005, November). Learning object reuse-a four tier model. In *People and Systems-Who Are We Designing For, 2005. The IEE and MOD HFI DTC Symposium on (Ref. No. 2005/11078)* (pp. 25-31). IET.

IEEE Learning Technology Standards Committee. (2002). Draft standard for learning object metadata. *Accessed July*, *14*, 2002.

Koehler, M. J., Mishra, P. (2009). What is technological pedagogical content knowledge. *Contemporary issues in technology and teacher education*, *9*(1), 60-70.

Krueger, C. W. (2013, August). Multistage configuration trees for managing product family trees. In *Proceedings of the 17th International Software Product Line Conference* (pp. 188-197). ACM.

Man, H., Jin, Q. (2010, April). Putting adaptive granularity and rich context into learning objects. In *Information Technology Based Higher Education and Training (ITHET), 2010 9th International Conference on* (pp. 140-145). IEEE.

McGreal, R. (2004). *Online education using learning objects*. Psychology Press.

Morales, R., Leeder, D., Director, U., Boyle, T., Director, L. (2005). A case in the design of generative learning objects (GLO): applied statistical methods GLOs. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*.

Polsani, P. R. (2006). Use and abuse of reusable learning objects. *Journal of Digital information*, *3*(4).

Premlatha, K. R., Geetha, T. V. (2015). Learning content design and learner adaptation for adaptive e-learning environment: a survey. *Artificial Intelligence Review*, *44*(4), 443-465.

Prieto-Diaz, R. (1988, January). Domain analysis for reusability. In *Software reuse: emerging technology* (pp. 347-353). IEEE Computer Society Press.

Rajlich, V. T., Bennett, K. H. (2000). A staged model for the software life cycle. *Computer*, *33*(7), 66-71.

Rossano, V., Joy, M., Roselli, T., Sutinen, E. (2005). A Taxonomy for Definitions and Applications of LOs: A Meta-analysis of ICALT papers. *JOURNAL OF EDUCATIONAL TECHNOLOGYAND SOCIETY*, *8*(4), 148.

Sametinger, J. (1997). *Software engineering with reusable components*. Springer Science & Business Media.

Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on* (pp. 85-90). IEEE.

Sosteric, M., Hesemeier, S. (2004). A first step towards a theory of learning objects. *Online education using learning objects*, 17-82.

Štuikys, V. (2015). *Smart Learning Objects for Smart Education in Computer Science: Theory, Methodology and Robot-Based Implementation*. Springer.

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

Štuikys, V., Damaševičius, R. (2013). *Meta-programming and model-driven meta-program development: principles, processes and techniques* (Vol. 5). Springer Science & Business Media.

Štuikys, V., Burbaitė, R., Bespalova, K., Ziberkas, G. (2016). Model-driven processes and tools to design robot-based generative learning objects for computer science education. *Science of Computer Programming*.

Stuikys, V., Burbaite, R., Bespalova, K., Drasute, V., Ziberkas, G., Venckauskas, A. (2016). Stage-Based Generative Learning Object Model to Support Automatic Content Generation and Adaptation. In Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual (Vol. 1, pp. 712-721). IEEE.

Taha, W. (1999). *Multi-stage programming: Its theory and applications* (Doctoral dissertation, Oregon Graduate Institute of Science and Technology).

Taha, W. (2004). A gentle introduction to multi-stage programming. In *Domain-Specific Program Generation* (pp. 30-50). Springer Berlin Heidelberg.

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

Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., Duval, E. (2012). Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies*, *5*(4), 318-335.

Wiley, D. A. (2000). Learning object design and sequencing theory.

Winograd, T. (2001). Architectures for context. *Human-Computer Interaction*, *16*(2), 401-419.

Yang, J. (2013). Research Guides: Instructor College: Instruction Resources: Instructional Strategies.

Zimmermann, A., Lorenz, A., Oppermann, R. (2007, August). An operational definition of context. In *International and Interdisciplinary Conference on Modeling and Using Context* (pp. 558-571). Springer Berlin Heidelberg.

Received January 31, 2017, revised May 3, 2017, accepted May 17, 2017