Informatics in Education, 2008, Vol. 7, No. 2, 277–288

Development of Generative Learning Objects Using Feature Diagrams and Generative Techniques

Vytautas ŠTUIKYS, Robertas DAMAŠEVIČIUS
Software Engineering Department, Kaunas University of Technology
Studentų 50, LT-51368, Kaunas, Lithuania
e-mail: vystu@if.ktu.lt, damarobe@soften.ktu.lt

Received: May 2008

Abstract. Learning Objects (LOs) play a key role for supporting eLearning. In general, however, the development of LOs remains a vague issue, because there is still no clearly defined and widely adopted LO specification and development methodology. We combined two technological paradigms (feature diagrams (FDs) and generative techniques) into a coherent methodology to enhance reusability and productivity in the development of LOs. FDs are used for knowledge representation, modelling variability of the LO content and relationships between its features, and as a high-level specification for generative reuse. The paper describes the specification of LOs using FDs and some design principles to design generative LOs.

Keywords: generative learning object, feature diagram, variability, learning object design.

1. Introduction

The concept 'learning object' (further LO or LOs) plays a key role in eLearning research. From the instructional design point, LOs are small, stand-alone, mediated, content "chunks" that can be reused in multiple instructional contents, serving as building blocks to develop lessons, modules, etc. (Wiley, 2000; Nugent *et al.*, 2006; Memmel *et al.*, 2007). When reused, such units are combined in various ways leading to a great variability of the learning content. The importance of the LOs is well understood; however, the understanding of the essence of the term is still poor (e.g., there is a variety of definitions of a LO (Rossano *et al.*, 2005; LTSC, 2002; McGreal, 2004; Polsani, 2004; Sosteric and Hesemeier, 2004), a variety of taxonomies and standards related to LOs (Rossano *et al.*, 2005; McGreal, 2004)).

Currently, LOs are a leading technology (LTSC, 2002) of choice for eLearning support due to its potential generativity, adaptability, and scalability (Wiley, 2000). LOs are also seen as computer-based teaching components that are to be modelled in the development stage in some well-established way. However, the development of LOs remains a vague issue, because there is still no clearly defined and widely adopted LO specification and development methodology as, e.g., in Software Engineering (SWE), where classes and objects are modelled using UML (Fowler, 2003). There have been several efforts to adopt UML for modelling interaction between LOs and a specific Learning Management

System (LMS) (Gao *et al.*, 2005), or to describe the content and process within "units of learning" in order to support reuse and interoperability (Laforcade, 2005). However, these efforts have not been entirely successful, because of inherent UML's orientation towards a very specific technological paradigm with specialized concepts such as 'concurrency' or 'polymorphism' (Friesen, 2004). Though there are efforts to extend UML to eLearning (Nodenot, 2007), variability cannot be easily expressed explicitly and modelled using UML, which has not adequate means for expressing variants of LOs.

In order to be delivered to learners, LOs must first be analyzed, specified, and represented. Specification of LOs plays a key role for the development, sharing and reuse. In such a context, we propose to use Feature Diagrams (FDs) for modelling (by modelling we mean the *conceptual modelling*) of the learning content. Originally, FDs were introduced in Feature-Oriented Domain Analysis (FODA) method (Kang *et al.*, 1990). Later, the idea was extended to software (SW) product lines (PLs) (Clements and Northrop, 2002). PLs, if used systematically, allow for dramatic increase of SW design quality, and productivity, provide a capability for mass customization and lead to the 'industrial' SW design (MacGregor, 2002).

The aim of the paper is to show benefits of these ideas to eLearning because the 'industry of LOs' can be specified, modelled and developed similarly to software components. Our contribution is the methodology based on using FDs and generative technology to develop generative LOs (GLOs). We have extended the GLO concept recently proposed by (Boyle *et al.*, 2004; Morales *et al.*, 2005), which can be seen as a PL for eLearning.

The structure of the paper is as follows. Section 2 considers basic concepts required to support feature-oriented modelling of LOs. Section 3 describes by an example the specification of LOs using FDs and delivers some generalizations. Section 4 presents some design principles to design GLOs using a generative technology. Section 5 analyzes a case study to support the introduced methodology. Section 6 presents evaluation (capabilities, usage) of the approach and conclusions.

2. Understanding of LOs through Analysis of LOs Features

Granularity, compositionality and reusability are key properties of LOs. However, there are numerous debates (Rossano *et al.*, 2005) on their actual meaning and of how reusability can be enhanced. Here we show of how these concepts and LO domain itself can be better understood by applying the feature analysis concepts (scope, communality and variability).

2.1. *What Are the Scope, Commonality and Variability of LOs?*

Let be given a set of objects (T) called LOs. Then the LO domain can be analyzed and categorized using concepts the scope (S), commonality (C), and variability (V). Such a categorization can be viewed as a result of SCV-analysis, the well-known activity in software engineering (Coplien *et al.*, 1998). This activity has been already introduced in the

eLearning, too (Diez *et al.*, 2007). We accept definitions of communality and variability proposed in (Coplien *et al.*, 1998). A *commonality* is an assumption held uniformly across a given set of objects T. Frequently, such assumptions are attributes with the same values for all elements of T. Conversely, variability is assumption true of only some elements of T or an attribute with different values for at least two elements of T. We consider Eq. (1) as a "simplest" LO only for better understanding of our concepts:

$$y = a \cdot x. \tag{1}$$

In Eq. (1), symbols '=' and '$\cdot$' and property 'equation has a left-side function notation and a right-side expression' is commonality. Here variability has much more dimensions, e.g., we can speak about *syntactic* variability because the content has a structure that can be described in various ways. Next, (1) can be represented graphically or as a table with the indicated values for x, a, and y. Furthermore, (1) has to be treated for different intervals of x (e.g., $[0, 1]$, $[-1, +1]$, or $[-\infty, +\infty]$). LOs have also a *semantic* variability because the content has the meaning and context. For (1), a semantic variability has to be conceived when one introduces various coefficients values for this equation (e.g., $a = 1$; $a > 1$; $a > 0$; $a < 0$) or yet another parameter b (e.g., b in $y = a \cdot x + b$ changes the structure (syntax) of the equation, while values of b (e.g., $b > 0$, $b < 0$) are influential to semantics).

As LOs are for learning (eLearning), they relate to various pedagogical theories and teaching scenarios, too (e.g., (1) can be delivered for students (pupils) as a time-distance relationship and be connected with travelling, say, during vocations). Learning as a social activity also depends on students' creativity, abilities, former knowledge, etc. Both constituents further extend the dimension of variability, which we call *social variability*. All kinds of variability is the *scope* of the LO *variability*.

In this paper, we consider the syntactic and semantic variability only. As it is clear from (1), variability appears along with communality constituting the whole structure of various relationships that form a LO and, therefore, both constituents are important attributes of a LO. However, the variability plays a specific role.

2.2. *Role of Variability for LOs Domain*

Variability means different variants of a LO leading to an explicit categorization of LO attributes ("*There is nothing more basic than categorization to our thought, perception, action, and speech*" (G. Lakoff)). However, it is not enough to categorize attributes; we need to discover *relationships* between different attributes. This relates to the *knowledge representation.* Recently, the ontology-based methods for representing knowledge [see, e.g., Proc. of ICALT '06, '07] became popular. Ontology is usually conceived as a *data model* that represents a set of concepts within a domain and *relationships* between the concepts. As the number of variants enlarges the scope of relationships, variability has a direct impact on the learning content.

Variability is also influential to granularity, compositionality and reusability of LOs, because today's technology enables us to create the content in a variety of versions.

A*daptations* are common reuse activities. The need for adaptation increases with technology advances and expansion of eLearning. If adaptations are done *ad hoc*, this may lead to the uncontrolled growth of similar versions causing extra difficulties in storing and reusing. If they are done automatically, we have a more powerful kind of reuse, *generative reuse*. GLOs (Boyle *et al.*, 2004; Morales *et al.*, 2005) are an example of such reuse. GLO provides more capabilities, focuses on quality issues, and introduces a solid basis for a marked improvement in productivity. GLOs are based on some representation of variants of LOs. The task is of how to capture and represent the variability explicitly?

2.3. *Domain Analysis of LOs and Representation Using Feature Diagrams (FDs)*

Kang *et al*. define *feature* as "end-user visible characteristic of a system or a key characteristic of a concept that is relevant to some stakeholder" (Kang *et al*., 1990). The intention of the concept is to represent a family of the related domain objects in some well-established way in order we could be able to model a domain through the relationships of feature variants. Feature modelling is the activity of modelling common and variable properties of concepts and their dependencies and organizing them into a coherent model referred to as a feature model. The model delivers the intention of a *concept*, whereas the set of instances it describes is referred to as an extension, which narrows the meaning and scope of the concept.

There are three types of features: *mandatory, optional* and *alternative*. Mandatory features allow expressing commonality of the concept, whereas optional and alternative features allow expressing variability. Features may appear either as a *solitary feature* or in groups. If all mandatory features in the group are derivates from the same father in the *parent-child* relationship there is the *and*–relationship among those features. An optional feature may be included or not if its father is included in the model. Alternative features, when they appear in groups as derivates from the same father, may have the following relationships: *or, xor* (filled arc in Fig. 1), *case* (arc in Fig. 1), etc. The *xor*-relationship also can be treated as a *constraint*. Usually a constraint appears when features are derived from different parents. For more advanced sub-types of alternative features as "views on ontologies", see (Czarnecki *et al.*, 2006).

A FD is a graphical notation of feature model, i.e., a tree-like or directed acyclic graph. The root represents the top level feature (i.e., concept, system or domain per se). The intermediate nodes represent *compound features* and leaves represent *atomic features* that are non-decomposable to smaller ones in a given context. Mandatory features are denoted as *boxes with black circles*, and alternative (optional) features as boxes with *white circles* (see Fig. 1). The edges are used to progressively decompose a compound feature into more detailed features. Edges of the graph also denote relationships between features nodes as it is depicted in Fig. 1.

As currently FDs is a hot topic of research (Schobbens *et al.*, 2006), there are many similar notations of FDs used. Our contribution is the explicit description of the contextual and functional relationships (see text in brackets besides a root in Fig. 1).

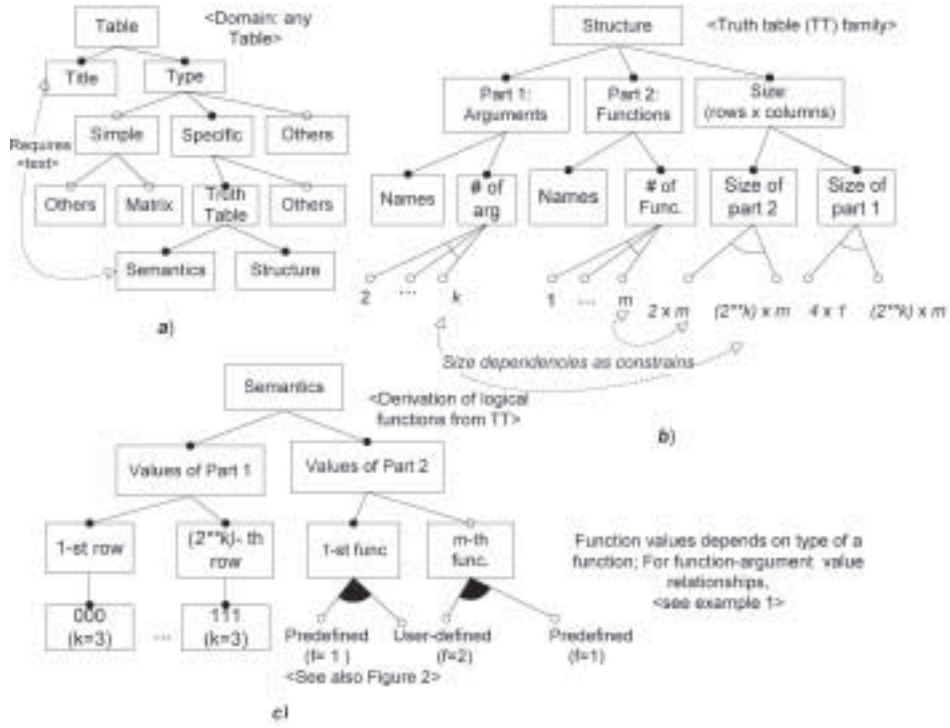

Fig. 1. Feature diagrams of learning object "Truth table" (a – general case, b – structure, and c – semantics of Truth table).

3. Specification of LOs Using Feature Diagrams: a Motivating Example

As a motivating example, we introduce the concept "Truth Table" (TT) that is widely used in Boolean algebra and digital systems design courses (Uyemura, 1999). By doing so, our aim is (1) to explain syntax and semantics of FDs and (2) to show of how the domain, i.e., all possible Tables (including TT) can be represented and modelled using feature concept and FDs. As the concept TT is a compound feature it is further decomposed into two mandatory features ('Structure' and 'Semantics', where the later means "content of the TT"). Since they have yet many sub-features and very rich semantic relationships, both are treated as separate LOs of the same subject. Leaves of the trees are atomic features that represent optional variants or values in parent-child relationships. There are also other types of relationships that are marked by doted lines; however, only a few of those relationships are shown (see Fig. 1 (a) and (b)). The most important relationships (see Eq. (2) and Eq. (3)) one can learn from Examples 1, 2.

Now we generalize the discussion and give an extended answer to the raised problems.

1. Because a FD describes LO features and their dependencies at a higher abstraction level, a FD is *a generic high-level model* for those LOs.
2. *Genericity* is expressed in the model through variability, i.e., through variants outlined by atomic features. For example, (see Fig. 1), one can estimate that part 1 of

TT has k and part 2 has $m \cdot 2^{2^k}$ variants.

3. The model describes all possible LOs, whose features and values of atomic features were included in the model. How much features are to be included and at what granularity level they are represented depends on goals of an analyzer.
4. Goals of the analyzer and the purpose of a LO are key attributes and they have to be reflected in FDs as a context of the use of the LO. We call this problem *contextualization* of FDs. Contextualization is an explicit representation of goals that means the context of using of FDs (see <*description of context*> at roots in Fig. 1).
5. *Granularity* depends on variability, type of features and the intention of a designer of how to group common and variable features with their relationships into a cluster to create the instance of a LO. Thus FDs can be seen as a tool for expressing (that means also measuring) of granularity explicitly.
6. *Compositionality* of a LO is also identifiable from FDs. As a FD is a decomposition of concepts by identifying their mutual relationships (also constrains) compositionality is seen as an opposite action to the decomposition. *References* (not shown) among FDs are a kind of the relationship that supports compositionality.
7. From the FDs model perspective, *reusability* is seen as a derivative property combining attributes mentioned above: granularity, compositionality and context of use. We discriminate two kinds of LO reuse: component-based and generative.
8. A FD is a tool for representing the LO knowledge as *ontology-like* trees.
9. A FD is a domain independent *high-level specification* for generative reuse. The specification is a vehicle to represent and model LOs. But this role is wider. It also provides the input information for automatic tools to generate LOs instances on demand. To be useful for automation such a specification has to be combined with generative technology. Thus, rephrasing the Willey's definition of a LO, we can say: *a FD is a concrete shape of a "chunk of the teaching content", with the identifiable level of granularity to support compositionality and reusability.*

4. Some Design Principles to Design Generative LOs

We describe these principles as a sequence of the following activities:

1. Selection and analysis of a domain of LOs.
2. Representation of the selected domain by a FD or by a set of FDs, i.e., the development of the high-level domain model.
3. Introduction of a relevant generative technology (e.g., templates, meta-programming (Sheard, 2001; Štuikys *et al.*, 2002) or various its kinds).
4. Selection of languages (e.g., meta- and target) for implementing, e.g., meta-programming.
5. The development of the model of the generative LOs. The model consists of meta-level for representing meta-data of a given domain and domain level.

6. Transformation of the high-level domain model (activity 2) and GLO model (activity 5) by merging them into a coherent model, using some transformation rules, which depends on the selected technology.
7. The result of the transformation is a GLO, which describes of how two models are combined into a coherent model using the selected technology. For meta-programming, the specification contains a meta-interface and meta-body. The first serves to specify metadata (types and values) extracted from the FDs model. Metadata are variants of atomic features to be managed during the generation session when a prescribed set of feature values are indicated to generate a LO on demand.
8. Verification of the developed specification.
9. Introducing (transferring) the verified specification into a web-based environment (server) in order to support eLearning.
10. On line verification of generative LOs through organizing sessions of links "server-PC" and testing of how the generated LO instances correspond to the metadata values and user requirements (*Note*: feedback is omitted).

5. Case Study

We demonstrate our ideas by presenting a simple GLO for explaining, specifying and using logic function truth tables for microelectronics education in MSc. level course "Reuse Technologies" lectured at Kaunas University of Technology (KTU). We distinguish between the metadata which is the metainterface of the GLO specification, and the instances of LO generated from the GLO. The role of the metainterface is to allow an educator to select the parameters of the GLO which reflect different features and variability of the GLO. Selection of the different values of these parameters can be used for LO personalization and tuning for different teaching tasks and student proficiency levels. GLO is usually accessible only for an educator and is stored on the LO server repository.

Fig. 2 presents the textual metainterface (metadata) of the GLO for a user (teacher, learner) to generate an instance of a Truth table LO on demand (see, e.g., a Truth table for 1-bit full adder shown in Table 1, here $x1, x2$ are the arguments; *sum, carry* are the functions; $k = 2$; $m = 2$; $f = 1$ are predefined values of metadata).

The metabody of the GLO is hidden from the end-user (because the technological details of meta-programming are important only for a GLO developer/programmer). We can add that the GLO can be implemented using a *meta-language* (Štuikys *et al.*, 2002)

```
"Select a number of arguments for Truth table"  {2..6}  k := 2;
"Select a number of functions for Truth table"  {1..3}  m := 2;
"Choice a mode for function(s)
           1 - predefined,
           2 - user-defined"                    {1, 2}  f := 1;
```

Fig. 2. Metainterface of generative learning object "Truth Table".

Table 1
Truth table of 1-bit full adder

x1	x2	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

that is used to describe and manage variability of LO instances described using a specific *domain language*.

The role of LO instances is to present a specific personalized learning content to the students. Therefore, students can access LO instances using an internet browser and can use it for learning course topics as well as for performing individual course tasks during course lab hours.

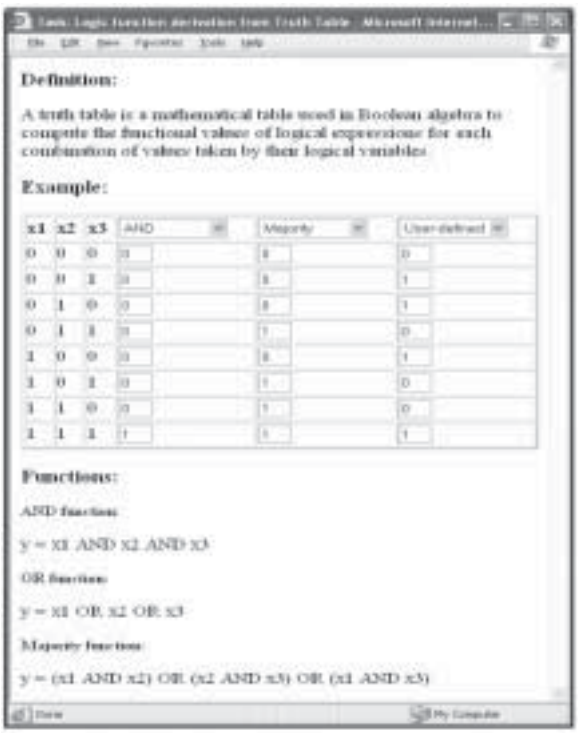

Fig. 3. Instance of learning object “Truth Table”.

EXAMPLE 1.

$$sum = (\textbf{not}(x1)\ \textbf{and}\ x2)\ \textbf{or}\ (x1\ \textbf{and not}\ (x2)) \tag{2}$$

EXAMPLE 2.

$$carry = x1\ \textbf{and}\ x2 \tag{3}$$

An example of an instance of the Truth Table LO generated from a GLO "Truth Table" with a graphical web-based interface is presented in Fig. 3. The LO introduces the student with basic definition of a truth table, and presents a simple example of truth tables for several predefined logic function. A student also can enter their own user-defined logic function into truth table and immediately obtain a corresponding logic equation, which is used later in the teaching process in implementing student hardware design projects. In this LO, two separate languages are used for expressing different aspects of LO: HTML is used for representation of learning content, and JavaScript is used for supporting interactivity and implementing functionality of the LO.

6. Discussion, Evaluation and Conclusions

Feature Diagrams (FDs) are tools serving for: 1) *designers, teachers and learners*, as a means for graphical representation of knowledge of the entire family of related learning objects (LOs) by providing domain ontologies using feature concepts, their types, values and relationships (Štuikys *et al.*, 2008); 2) *programmers*, to specify and express variability-communality relationships of LOs at a higher abstraction level in order to develop and implement generative LOs (GLOs) systematically; 3) *researchers and other actors*, as a vehicle for analysis and better understanding of the LOs domain itself because FDs enable to express granularity, compositionality, and context explicitly to support reusability.

A GLO is a specification describing a family of the related LO instances. It has the user manageable metadata for deriving instances on demand. A particular GLO can be seen as a *mini repository* of LOs providing the possibility to automatically generate from the repository a concrete LO instance on demand depending on the metadata values that user identifies. A GLO may be a member of a conventional LOs repository too, but each LO instance of GLO must be first generated before using.

LO instances derived from GLO make up only a part of a topic (course). They are integrated with other components (LOs) in order to create a LO of the higher granularity level. They are extremely useful for Lab works, for problem solving and design tasks. Benefits of using GLOs are evident especially in those cases where time-consuming and error prone activities arise in eLearning as it was demonstrated in our case study (and also in eLearning on demand (Schmidt, 2007)). We use GLOs supplemented with FDs in various Computer Science (CS) oriented courses to demonstrate IT capabilities and generative reuse to enhance eLearning and learning.

The discussed topics, FDs and GLOs, are new enough for eLearning. Basic ideas to construct our discussion were borrowed from SWE (CS). Our contribution is the adaptation and extension of these ideas to the LOs domain. However, further research is needed to exploit the full potential of FDs and GLOs in the domain.

References

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

Clements, P. and Northrop, L. (2002). *Software Product Lines: Practices and Patterns.* Addison-Wesley.

Coplien, J., Hoffman, D. and Weiss, D. (1998). Commonality and variability in software engineering. *IEEE Software*, **15**, 37–45.

Czarnecki, K., Kim, C.H.P. and Kalleberg, K.T. (2006). Feature models are views on ontologies. In *Software Product Line Conference*. Baltimore, USA, August 21–24.

Díez, D., Fernández, C. and Dodero, J.M. (2007). An effective analysis method for computer-supported learning systems reusability. In *IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables* (*SPDECE07*). Bilbao, Spain, October 19–21.

Fowler, M. (2003). *UML Distilled*: *A Brief Guide to the Standard Object Modeling Language.* Addison-Wesley.

Friesen, N. (2004). Three objections to learning objects and e-learning standards. In R. McGreal (Ed.), *Online Education Using Learning Objects*. Routledge, 59–70.

Gao, J., Marchetti, E. and Polini, A. (2005). Applying advanced UML based testing methodology to e-learning. In *Proc. of the IADIS Int. Conf. on Applied Computing.* Algarve, Portugal, February 22–25, 74–79.

Kang, K., Cohen, S., Hess, J., Novak, W. and Peterson, S. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University.

Laforcade, P. (2005). Towards a UML-based educational modeling language. In *Fifth IEEE Int. Conf. on Advanced Learning Technologies, ICALT 2005*. 855–859.

Learning Technology Standards Committee (2002). *IEEE Standard for Learning Object Metadata*. IEEE Standard 1484.12.1. IEEE, N.Y.

MacGregor, J. (2002). Requirements engineering in industrial product lines. In *Proc. of Int. Workshop on Requirements Engineering for Product Lines REPL'02*. Essen, Germany, 5–11.

McGreal, R. (Ed.) (2004). *Online Education Using Learning Objects.* Routledge, N.Y.

Memmel, M., Ras, E., Jantke, K. and Yacci, M. (2007). Approaches to learning object oriented instructional design. In A. Koohang and K. Harman (Eds.), *Learning Objects and Instructional Design*. Informing Science Press, 281–326.

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

Nodenot, T. (2007). From UML to CPM: a few lessons learnt about CPM language usability. In *Proc. of 7th IEEE Int. Conf. on Advanced Learning Technologies* (*ICALT 2007*).

Nugent, G., Soh, L.-K. and Samal, A. (2006). Design, development, and validation of learning objects. *J. Educational Technology Systems*, **34**(3), 271–281.

Polsani, P.R. (2004). Signs and objects. Modelling learning objects on Peirce's theory of signs. In R. McGreal (Ed.), *Online Education Using Learning Objects*. Routledge.

Rossano, V., Joy, M., Rosselli, T. and Sutinen, E. (2005). A taxonomy for definitions and applications of LOs: A meta-analysis of ICALT papers. *J. Educational Technology and Society*, **8**(4), 148–160.

Schmidt, A. (2007). Enabling learning on demand in semantic work environments. In J. Rech, B. Decker and E. Ras (Eds.), *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications.* IGI Publishing.

Schobbens, P.-Y., Heymans, P., Trigaux, J.-Ch. and Bontemps, Y. (2006). Feature diagrams: a survey and a formal semantics. In *14th IEEE International Requirements Engineering Conference* (*RE'06*).

Sheard, T. (2001). Accomplishments and research challenges in meta-programming. In *2nd Int. Workshop on Semantics, Application, and Implementation of Program Generation* (*SAIG'2001*). Florence, Italy. *LNCS*, vol. 2196. Springer, 2–44.

Sosteric, M. and Hesemeier, S. (2004). A first step towards a theory of learning objects. In R. McGreal (Ed.), *Online Education Using Learning Objects.* Routledge, N.Y.

Štuikys, V., Damaševičius, R. and Ziberkas, G. (2002). Open PROMOL: an experimental language for target program modification. In A. Mignotte, E. Vilar and L.S. Spruiell (Eds.), *System-on-Chip Design Languages.* Kluwer Academic Publisher.

Štuikys, V., Damaševičius, R., Brauklytė, I. and Limanauskienė, V. (2008). Exploration of learning object ontologies using feature diagrams. In *Proc. of World Conference on Educational Multimedia, Hypermedia and Telecommunications* (*Ed-MEDIA 2008*). June 30–July 4, Vienna, Austria. Chesapeake, VA: AACE, 2144–2154.

Uyemura, J.P. (1999). *A First Course in Digital Systems Design*: *An Integrated Approach*. An International Thomson Publishing Company.

Wiley, D.A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D.A. Wiley (Ed.), *The Instructional Use of Learning Objects*.

Wiley, D.A. (2000). *Learning Object Design and Sequencing Theory*. PhD Thesis, Department of Instructional Psychology and Technology, Brigham Young University.

V. Štuikys received PhD degree from Kaunas Polytechnic Institute in 1970 and doctor habilitatis from Kaunas University of Technology in information technology in 2003. Currently he is a professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interests include domain-specific and software reuse, high level domain-specific languages, component-based programming, metaprogramming and program generation, expert systems and CAD systems, including eLearning systems, and soft IP design.

R. Damaševičius received PhD degree in 2005 in informatics from Kaunas University of Technology, Kaunas, Lithuania. Currently he is an assoc. prof. at Software Engineering Department, Kaunas University of Technology. His research interests include metaprogramming, software reuse, software generation and program transformation, as well as domain analysis methods.

Generatyvinių mokymo(-si) objektų kūrimas panaudojant požymių diagramas ir generavimo metodus

Vytautas ŠTUIKYS, Robertas DAMAŠEVIČIUS

Mokymo(-si) objektai yra esminis e-mokymo elementas. Nepaisant to, mokymo(-si) objektų kūrimas vis dar išlieka problematiškas, nes iki šiol nėra apibrėžtos ir plačiai taikomos mokymo(-si) objektų specifikavimo ir projektavimo metodikos. Siekdami padidinti mokymo(-si) objektų atkartojamumą ir projektavimo našumą mes apjungėme dvejas technologijas: požymių diagramas ir generavimo metodus. Požymių diagramos yra naudojamos žinių atvaizdavimui, mokymo(-si) objektų variantiškumo ir ryšių tarp jų požymių modeliavimui ir kaip aukšto lygmens specifikacija. Straipsnyje aprašomas mokymo(-si) objektų specifikavimas naudojant požymių diagramas ir pateikiami kai kurie generatyvinių mokymo(-si) objektų projektavimo principai.