ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol.36, No.2

TOWARDS KNOWLEDGE-BASED GENERATIVE LEARNING OBJECTS

Vytautas Štuikys, Robertas Damaševičius

Software Engineering Department, Kaunas University of Technology
Studentų St. 50, LT–51368 Kaunas, Lithuania

Abstract. Today there are many efforts to shift the reuse dimension from component-based to generative reuse in the learning object (LO) domain. This requires more precise LO models and commonality-variability analysis. We propose a new knowledge-based model for representing LO instances. The model is based on factoring and aggregating knowledge units within a LO and is presented as a structure of interface and functionality. Interface serves for explicit describing knowledge communication to and from the LO. Functionality describes knowledge representation and managing. The model contributes to better compositionality, reusability and can be further generalized easily to support the personalized content delivery and automatic generation. Using the introduced model as a basis for generalization, we extended the known concept of generative LOs by linking domain commonality-variability analysis with meta-programming techniques for generating LO instances on demand from the generic LO specification.

Keywords: learning object model, generative learning object, commonality-variability analysis.

1. Introduction

1.1. Motivation for LOs

The term *learning object* is the most essential concept in the learning domain due to many reasons:

- a learner needs to know *why* and *what* to learn;
- a teacher needs to know and understand *why, what* to teach and *how* to deliver the content to the learner;
- a course designer needs to know and understand *what* to design and *how* to represent the content for storing and sharing;
- curricula developers also need to know and understand *why, what* and *how* to plan and additionally – *where* and *when* – the courses are integrated into teaching plans.

Those and many other actors, such as researchers, use the learning content in the sense of *what* (i.e. in the sense of LOs) either as an object of investigation or as commodity in their daily activities. This is why we need to have a measure for expressing and measuring the learning content as precisely as possible.

1.2. Reuse and LOs

Despite the numerous and continuous efforts to introduce taxonomies and standards by bodies (e.g., IMS, IEEE LTSC, ISO/IES) or individuals in this area, the term *learning object (LO)* is neither understood well nor is defined uniformly. A variety of LO definitions and models exist and new proposals appear continuously (see e.g., [1]). What is more or less commonly agreed by most actors within the community are the *role of reuse* and the potential of *reusability,* because the same or similar content is delivered in many different contexts at numerous teaching institutions repeatedly and continuously worldwide. Though for long time reusability was in the focus of researchers, so far reusability was understood narrowly, mostly in terms of *component-based* reuse. However, in recent years there are noticeable endeavors to enhance reusability by introducing novel approaches, such as glass-box LOs [2], aspect-oriented LOs [3], adaptive LOs [4] or LOs based on object programming concepts [5].

1.3. Generative reuse and generative LOs

Today technology advances enable teachers and course designers to create the content in a variety of versions. Modifications, changes and adaptations of the content are common reuse activities. The need for adaptation increases with technology advances and expansion of the eLearning domain. If adaptations are done ad hoc, this may lead to the uncontrolled growth of similar versions causing additional difficulties in storing, sharing and reusing. If adaptations can be done automatically, we have a more powerful kind of reuse, called *generative reuse*. Recently T. Boyle, D. Leeder, *et al*. [6, 7] have proposed the concept of generative learning objects (GLOs), which is based on separating the learning design from the instantiation of the LO content and using templates as a generative technology. The approach provides more capabilities at a larger extent, focuses on quality issues, and intro-

duces a solid basis for a marked improvement in productivity.

To extend the reusability dimension further in the LO domain, we need a more attentive look at the domain itself from the reuse perspective; or more precisely, we need to analyze such LOs features as *commonality* and *variability* (see Section 4) systematically. The first relates more with component-based reuse, while the second may be a solid background for generative reuse, if obtained and represented adequately. Though the learning theories [8, 9] actually recognize and consider many features that might be conceived as commonality and variability, they emphasize the pedagogical or psychological viewpoint only, without explicit representation of variability and any intent in using generative technologies explicitly.

1.4. Our approach

Our approach is based on commonality-variability analysis of the related LOs and is aiming to extend the concept of GLOs already introduced in the domain. As LOs represent the content that may vary across different courses and the delivery of the content relates also to pedagogical aspects (e.g., motivation, scenarios, teaching theories, etc.), social aspects (e.g., teachers' preferences, students' abilities, collaborative eLearning, self-learning, etc.) and technological aspects (e.g., representation in eLearning, or in mLearning, etc.), the boundaries of variability may be extremely large. Thus we need to introduce some restrictions on granularity and type of LOs, and on scope of variability. We accept the middle-grained or fine-grained level of granularity [10] that could form the content of a few lectures/lessons at largest, or represent one lesson or be a part of that (e.g., sorting algorithms in computer science or properties (laws) of Boolean algebra, etc.). Some ideas we have incorporated in our approach are borrowed from software engineering or computer science (e.g., SCV-analysis [11], explicit representation of interface and functionality in the LO instance model [12], heterogeneous meta-programming techniques [13]), but they are extended and adapted here. As the generative approach mainly focuses on the representation aspects (e.g., LO domain as a family of related LO instances and an instance per se), we have reviewed and re-evaluated the known LO models and, as a result of that, proposed the knowledge-based model.

1.5. Novelty and tasks

We propose a new knowledge-based model for representing LO instances. The model is based on factoring and aggregating knowledge units within a LO and is presented as a structure of *interface* (for knowledge communication in both directions: *to* the LO and *from* the LO) and its *functionality* (for knowledge representation and managing). The model contributes to better compositionality, understandability, extensibility, reusability and can be further generalized easily to support the personalized content delivery and automatic generation. In particular, we extend the known concept of GLOs by connecting commonality-variability analysis in the domain with heterogeneous meta-programming techniques for generating LO instances on demand from the generic LO specification.

As a consequence, we formulate and consider the following tasks:

1. Description of the knowledge-based LO instance model (Section 3);b
2. Description of the approach for analysis of commonality and variability in the LO domain (Section 4);
3. Description of the generative LO model and a framework for its implementation using heterogeneous meta-programming techniques (Section 5).

Section 6 summaries and evaluates capabilities of the approach and also presents indications on some limitations and tasks for the further work. At the end, conclusions are formulated.

2. Definition of the basic terms

First, we begin with the definition of the basic terms. We accept the following scheme for definitions:

1. <Name of term>: ***mandatory***
2. <Intention>: *optional*
3. <Definition>: ***mandatory***
4. <Consequence(s)>: *optional*

Note that all attributes are mandatory for basic terms defined below; however for other terms defined within Sections some attributes (e.g., No.2, No.4) may be optional.

1.1**: Learning Object;**

1.2: *To split learning content into parts and treat them uniformly*; *or to compose the content from the parts;*

1.3: LO is "*a form of organized knowledge content … involving learning purpose and reusable value*" (P.R. Polsani) and, additionally, its granularity level is restricted to the fine-grained or middle grained component [10];

1.4: A middle-grained component is composed of the fine-grained LOs.

2.1**: LO structure (model);**

2.2: *To enhance reusability of LO through factoring of knowledge and indicating the role of explicit separation of LO interface from its internal functionality*;

2.3: A structure describing the interface for communicating knowledge with other LOs (or among teacher and student) and the functionality for representing internal relationships between knowledge units and managing knowledge;

2.4: Easiness for composition or aggregation of a larger LO, better understandability and reusability of LOs.

3.1**: LO instance;**

3.2**:** *To use in a concrete context*;

3.3: A detailed description of the LO model containing concrete objectives and delivering knowledge for the given context;

3.4: Use in a LO library, use for composition and design of a new course, etc.

4.1**: Learning Object interface;**

4.2: *Explicit description of communication and compositional aspects of a LO, for example for transferring essential knowledge that is needed for learning and creating knowledge to be learned*;

4.3: A structural list of input knowledge and a structural list of output knowledge;

4.4: Contribution to managing flexibility and better learners' guiding.

5.1**: Learning Object functionality;**

5.2**:** *To provide a detailed description (instantiation) of the internal implementation of a LO and produce output knowledge***;**

5.3**:** A detailed description of functional or structural implementation of LO functionality through various kinds of relationships applied to knowledge units;

5.4: A full description of LO (in sense of stated objectives for the LO).

6.1**: Basic knowledge unit;**

6.2: *To construct aggregated knowledge from atomic knowledge units;*

6.3: Atomic knowledge unit that consists of *container* and *content* both of which cannot be decomposed in smaller parts; but also decomposable knowledge, if already obtained by learner, it may be treated in the other context as atomic.

6.4: A wide context of the use and knowledge composition/aggregation.

7.1**: Aggregated knowledge;**

7.2*: To provide composition and sequencing, to extend the level of knowledge representation.*

7.3: Knowledge derived through the aggregation of basic knowledge units or other aggregated knowledge by applying a particular kind of relationships.

7.4: Knowledge composition.

8.1**: Input knowledge;**

8.2**:** *To define the input part of the LO interface;*

8.3: Basic or aggregated knowledge unit in the input list of the LO interface coming from the other LO (e.g., previous lecture) which is used to explain and understand the formation of new knowledge.

8.4: Contribution to better managing and aggregated knowledge creating.

9.1: **Output knowledge;**

9.2: *To define the output part of the LO interface;*

9.3: A basic or aggregated knowledge unit in the output list of the LO interface, which is identified within the functionality description of the given LO and which is aiming at being learnt and transferred to the other external LO or learning environments (e.g., previous lecture, other course, etc.).

9.4: Contribution to better managing, composing and reusing.

10.1**: Generative LO;**
10.2: *To introduce a generative technology for automatic generation of LO instances;*
10.3**:** A family of related LO instances that are packaged into a higher-level specification (meta specification) using some generative technology such as meta-programming; depending on the concrete context of use, the user specifies the needed values of parameters and a particular LO instance is derived or generated on demand from the specification automatically.
10.4: Extension of the reuse dimension and basis for higher productivity.

3. Description of LO instance model

3.1. LO and knowledge concept

The below stated is an introduction to the known theories, such as instructional design [8, 9], basic concepts of which are incorporated in our LO model. The intention of the LO model is to describe a scheme for representing *knowledge explicitly.* There is no single definition of the term *knowledge*. Our understanding of the term is similar to T.H. Davenport's [14] and is based on the *data-information-knowledge* relationship. Instead *information*, we use the term *data context*. Thus our understanding of knowledge is the chain of transformations and engagement: *data → data plus its context → knowledge*. We present an example to explain the difference between data and knowledge. The set of symbols {Λ, AND, &} without a context is *pure data for a novice learner*. However, the explanatory text "*symbols denote the logic operation (conjunction) in different Boolean algebra notations"* contributes to transforming data into knowledge, perhaps, by introducing other data via the learning process. From the pure representation perspective, knowledge has *container* and *content*. For example, the statement "*The Second Newton's law*" is the container, and the statement "*The force of an object is equal to its mass times its acceleration, i.e. F= m*a*" is the content.

The context of data/knowledge can be given explicitly (e.g., through explanation, examples, etc.) or implicitly. This *is a weak form* of knowledge since it focuses on the representation view only. Knowledge that resides in human minds relates to *understanding the meaning*. This is a *strong form* of knowledge. When a teacher represents a LO, he/she represents data about the LO and its context. The understanding, i.e., the strong form of knowledge comes through the learning process of learners (see, for example, [2] and [14, 93 p.]). As we deal here with the representation aspects of LOs we manipulate the weak form of knowledge. The context of a LO is the most essential part because it brings the potential for understanding, the *data transformation* into knowledge.

3.2. Categories of knowledge

As knowledge has a direct relationship to data we can admit that knowledge is composed of other knowledge using some operations, which we call *associations or aggregations* in this paper. Associations are some *relationships* defined on "simpler knowledge". We define two basic categories of knowledge: *atomic knowledge unit* and *aggregated knowledge*. The first either cannot be decomposed into smaller parts (e.g., knowledge that contains the definition of the term "conjunction" operation in Boolean algebra) or learner/teacher treats it as indivisible item in the given context, though in the other it might be divisible (e.g., a sort algorithm in the Data Base course may be the atomic knowledge unit, while in the Computer Science course not).

Aggregated knowledge is composed either of atomic knowledge units or of other "lower-level" aggregated knowledge. Any kind of knowledge can be grouped and sequenced into clusters. Such a cluster is aggregated knowledge too. Aggregated knowledge, if it used for many times in different places of the LO, can also be seen as an internal component. In different subjects, we can speak about clusters of notations, clusters of terms, etc. When new aggregated knowledge is represented within a LO, the new context to this knowledge is to be obtained too.

3.3. LO model structure

The model consists of three basic parts: *name, knowledge-based interface and knowledge-based body*. The first, such as the topic/theme name, is for identification and referencing or the learning objective statement (e.g., as the context of the name). Interface is for communicating and transferring knowledge to the LO and from it. As teacher and learner communicate knowledge, interface can be seen also as a media for the teacher/learner interaction, or as a media learner/learner interaction in the case of self-learning. Interface, in turn, consists of two kinds of knowledge: *input* and *output knowledge*. Interface is clearly separated from the body. Input knowledge is a structured (sequenced) list of atomic knowledge units coming from outside. Output knowledge is a structured list of aggregated knowledge identified within the body, which is to be learned and then transferred to other LOs or the learner. From the learner's perspective, output knowledge is shift in time with respect to input knowledge within a given learning process when it is initiated. Note that conceptually input knowledge is called 'prerequisites' in pedagogy theories [8]. The difference is that in our case this knowledge is embedded in the structure explicitly. By input knowledge, we mean the knowledge which a component accepts as its input. By output knowledge, we mean the knowledge which a component produces as output while associations are applied to the input knowledge. By

internal knowledge, we mean the knowledge which is important for building output knowledge but is used only within a component and is not transferred externally.

3.4. Properties of knowledge-based interface

The properties of the interface are:

- Input/output knowledge are transferred in the *in/out* mode, respectively;
- Output is shifted in time with respect to input while the LO is processed in the learning process (learner's view);
- Input has the *implicit (default)* context because it is already learnt knowledge and its *explicit* context is "left" in the previously learnt LO; in some cases (e.g., in self-learning, repeating material) input knowledge may be omitted;
- Output may have or not the explicitly stated context in the interface. This depends on the designer's intention;
- Typically input knowledge depends on learning objectives and has to be consistent with learning goals and the current learner's knowledge. If not, internal knowledge should be declared within the body in order to resolve inconsistency;
- Explicit interface supports external composition of LOs into higher-level structures;
- Clearly stated objectives restrict the scope of input and output knowledge, and designer reasons about the scope, especially about the input knowledge;
- Quality expert reasons about consistency of input knowledge with goals and quality of the LO in a whole when the model is being instantiated.

3.5. Knowledge-based body

The body defines implementation details of a LO, i.e. structural and functional aspects. The body contains a few important sections: *declarative, procedural, contextual and managerial* (Figure 1). In declarative section, one can state the objectives of the LO (if this statement was omitted in the context of the LO name), identify scenarios to be applied in learning, references to context (e.g., examples, case studies, or material for refreshing input knowledge in ones memory), auxiliary data (e.g., internal knowledge to support consistency of input knowledge with learning objectives).

The procedural section presents all kind of associations, which provide a means to produce output knowledge. These associations may be formed of input knowledge only or be constructed from the input and internal knowledge or already constructed knowledge. As input knowledge is structured in two different forms (structured pieces of data and structured pieces of context though it is defined by default, i.e. implicitly), output knowledge is produced in the structural way too. The associations can be described textually, graphically or in the abstract or formal way.

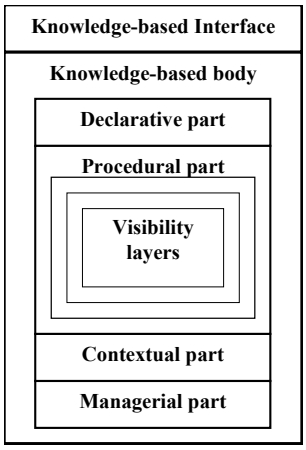

Figure 1. Structure of LO model

The contextual part contains explanations, test cases, self-learning objects, examples, references for further readings, etc. Some learning objects (knowledge) in that part may be teacher-oriented (e.g., tasks for test, answers, etc.) while the other objects are student-oriented. What is common to all the above mentioned structural units of the model is that within each section the information is strongly clustered into blocks, each block is being labeled to enhance manageability.

The managerial section serves for providing control information to manage various aspects of the model usage. The aspects, for example, may include:

- Quality aspects (expert's view)
- Teacher's view to management tasks
- Student's view to management tasks (self-learning)
- Aspects of the representation component in digital (web-based) environment
- Aspects for implementing navigation within the given LO component sections (e.g. in case of self-learning), among the different components of a given course or navigation between different courses.

Implementation of the above stated functionality is based on the two basic principles: granularity of information within sections and information hiding. For example, some information is not available for students and is available for teachers only. In other cases, one may want to deal with only interface of a component, without its body (e.g., in self-learning or repeating).

The LO model describes the structural and functional aspects (associations and output knowledge formation principle). In order to solve the representation problem, we need to have some notation or language. Various kinds of languages can be chosen to implement the task. We consider several choices: UML, XML and the narrative natural language (English). In

our view, the latter is the best for the purposes of course planning. Below, we present an example of the component model in the natural language. For readability, the format is supplemented with extra words and separators, which have the following meaning and in the other context may be omitted:

- Bold - important term of the model
- Bold and italic – important term related to LO
- Italic – name of knowledge fragment (unit) or associative knowledge
- (…) – context
- [*x*] – reference to the external source *x*
- Label: **begin**… **end***;* - block of knowledge fragments (units of the LO); but they are optional (the label is obligatory).

3.6. LO model example with its instantiation

Name of LO: T1. ***Boolean algebra principles for restricted set of operations and arguments.***
(*Goal is to explain the Boolean algebra principles and laws for homogeneous functions)*
Interface to T1 **is**
Begin
Input knowledge is
L1: begin *Operations "*" and "+" in algebra and arguments-operations functional relationship* **end;**
L2: begin *Definition of the range of values for arguments and algebraic functions* **end;**
Output knowledge is
L3: begin *Definitions of Boolean functions "AND", "OR" in the narrative and truth table forms* **end;**
L4: begin *Commutativity law* **end;**
L5: begin *Associativity law* **end;**
end;

Knowledge body of T1 **is**
begin
Declarative part is
L6: begin *Scenarios: learning by examples* **end**;
L7: begin *By analogy with algebra* (see L1 and L2)*, Boolean functions and their arguments have value ranges, but what is different is that these values vary in the restricted interval {**true, false**}* **end**;
L8: begin *Spelling and notations for functions:* ***disjunction*** *"OR";* ***conjunction*** *"AND"* **end;**
Procedural part is
L9: begin *Teacher/learner (T/L) selects a Boolean function and introduces definitions:*
"Logical conjunction is the function that results in the value of ***true*** *if all its arguments are* ***true****, otherwise a value of* ***false****."* (see L12)
"Logical disjunction is the function that results in the value of ***true*** *just whenever some of its arguments are* ***true****"* (see L13) **end;**
L10: begin *T/L selects a Boolean function and demonstrates* ***commutativity law*** (see L14) **end;**
L11: begin *T/L selects a Boolean function number of arguments equal to 3 and demonstrates* ***associativity law*** (see L15) **end;**

Contextual part is
L12: begin *Truth table* for ***conjunction*** **end;**
L13: begin *Truth table* for ***disjunction*** **end;**
L14: begin *Y= a AND b* ***is equivalent*** *to Y= b AND a; X= a OR b* ***is equivalent*** *to X= b OR a,*
where *X, Y, a, b* = {***true, false***} **end;**
L15: begin *((a AND b) AND c) = ((a AND c) AND b); ((a OR b) OR c) = ((a OR c) OR b),*
where *a, b, c* = {***true, false***} **end;**

Managerial part is
L16: Teacher's view: *Teacher formulates problems: 1."To construct truth table with 3 and 4 arguments for two groups of students" (teacher sends templates of truth tables or explains how they should be generated automatically* [e.g., look at web site http://xxx]*) 2. Teacher asks to calculate value of a given function.*
L17: Student's view: *Students fill in truth tables and sent the result to teacher (via Internet) for evaluation*
End body;

Notes: 1) this is a fine-grained LO to learn *homogeneous Boolean functions*; by using the described framework and introducing an additional operation/function (e.g., NOT), a higher-level LO for the general case, i.e. to learn *heterogeneous Boolean functions*,

can be constructed easily; 2) truth tables are not shown; for a case, see Figure 4; 3) xxx – virtual site.

4. Commonality-variability analysis and domain model creation for the related LOs

"*There is nothing more basic than categorization to our thought, perception, action, and speech*" (George Lakoff [14, 144 p.]).

4.1. Definition of commonality and variability

The LO domain can be analyzed and then categorized using such concepts as scope (S), commonality (C), and variability (V). Such a categorization can be viewed as a result of SCV-analysis. We accept definitions of commonality and variability proposed by J. Coplien and his colleagues in [11]. A *commonality* is an assumption held uniformly across a given set of objects (S). Frequently, such assumptions are attributes with the same values for all elements of S. Conversely, a *variability* is an assumption true of only some elements of S, or an attribute with different values for at least two elements of S.

We illustrate these concepts with a simple example taken from the Boolean algebra domain. Let us consider S being the set of all "homogeneous" AND, OR and NOT functions (Λ, V and ¬ are other notations for the functions, respectively). The value of a function can be treated as output and its arguments as inputs. The attributes "any function has one output", "input/output have names", and "input-output relationship is expressed by Boolean equations" are commonalities (see also (1) and (2)). The attributes "type of function" or "number of inputs for homogeneous AND-, OR-functions, which may vary from two to any" are variability (e.g., compare (1) and (3)). The attribute "NOT-function has only one input" is specificity (see also (4)).

y = x1 **AND** x2 **AND** x3; (1)
y = x1 **OR** x2 **OR** x3; (2)
y = x1 **OR** x2; (3)
y = **NOT**(x1); (4)

4.2. Factoring LOs domain

SCV- analysis may result in factoring of the LO domain as follows:

- S is the set of *all related* LOs driven by a given curricula or constrained by a given course or themes within the course;
- C is the characteristics common to all members of LOs in S;
- V is the variation of the content among LO instances as well the variation of other aspects (e.g., pedagogical, social, technological) but these should be considered separately;
- P is the features of specificity (if any); and
- I is the relationship or *interaction* among C, V (P is excluded due to it should always be isolated) in S.

The set S introduces the boundaries in analysis. However, the boundaries can be either broadened or narrowed by the analyst (e.g., instructional designer) depending on goals and given resources. For example, she or he can connect analysis of the content with analysis of known models for representing of LOs or learning scenarios. C is most influential to reuse in the mode "use-as-is", while V is most influential to automation, i.e. generative reuse. We call interaction the perception of the role C in respect to the role of V in the given context. For example, the interaction can be viewed as understanding the same role of different things in the same context; or vice versa, understanding the different role of the same things in the similar or, perhaps, the same context. Let us consider two strings: 'x1, x2, x3' and 'x1 or x2 or x3'. The different symbols ',' and 'or' (variability) have the same role: they both are separators (commonality).

4.3. Creating model for the domain of related LOs

What we need for the generic specification of the related LOs is the explicit representation of analysis results, i.e. synthesis of the explicit model of the domain. From this perspective, variability is to be expressed through parameters with clearly stated values and their ranges (within the given scope). The short description of scenarios of relationships is very helpful too. Referencing to our example as illustrative, we present the extended model for the LO "Homogeneous Boolean Equations" explicitly (see Table 1). The model is called commonality-variability model and describes the whole domain of the LO from which concrete instances can be derived on demand depending on parameters' values. When instances are created, the LO model such as described in previous Section 3 is used for representing the concrete content. Note that we restrict ourselves by analysis of the content only.

The described analysis and synthesis processes can be viewed as *self-learning of the analyst per se* due to: 1) the objectives and scope are pre-specified; 2) the sequence of actions (though not so much in detail) is described; 3) the result is explicit but depends on many factors such as skill, previous knowledge and experience of the analyst; 4) the processes may require revision and improvement, or perhaps, interaction with experts; 5) the synthesized model is to be tested for quality.

To achieve our goals in creation of GLOs, the proposed models (LO model and C-V model for the given class of LOs) are to be connected to the appropriate generative technology, such as meta-programming [13].

Table 1. Commonality-variability model for LO "Homogeneous Boolean functions"

LO parts after factoring	Possible LO features and their values	Scenarios
Commonality (C)	1. Output-input functional relationship 2. Range of Inputs/Output values 3. Definition of a function 4. Statement of commutative law for AND-, OR-functions 5. Statement of associative law for AND-, OR-functions	y = x1 **AND** x2**;** y**={true, false};** x1**,** x2**= {true, false}** y = x2 **AND** x1↔ y = x1 **AND** x2 y = (x1 **OR** x2) **OR** x3↔ y = x1 **OR** (x2 **OR** x3)
Variability (V)	1. *Kinds (k)* of definitions to be learnt 2. Kinds of *laws* (*l*) to be learnt 3. *Types* of functions (*f*) 4. *Number* (*n*) of inputs for any function 5. Notations of functions We ignore other kinds of variability (e.g., names)	Textual (Narrative) / Truth table Commutative /Associative y = x1 **OR** x2; y = x1 **AND** x2**;** y = x1 **OR** x2 **OR** x3; (for AND: Λ, & for OR: V, |)
C-V interaction	1. V1-C1-C2; 2. V2- C4-C5	Knowledge about interaction are implicit
Interaction within variability	1. V4 interacts with V1 2. V4 interacts with V2	Knowledge about interaction are implicit
Scope (S) of parameters variation	1. *f* = {AND, OR } 2. *n* = {2, 3, 4, …, 16} 3. *k* = {textual, table} 4. *l* = {commutative, associative}	S = |f|×|n|×|k|×|l| = 2×15×2×2 = 120 (Scope of the whole LO domain: "homogeneous Boolean equations" planned for learning)
Specificity (Sp)	**1. NOT- function has only *one input***	y = (**NOT**) x1;

5. Development of GLOs using heterogeneous meta-programming techniques

5.1. Introduction to meta-programming

In general, meta-programming is defined as *a manipulation with programs as data* [13]. Meta-programming can be viewed as a *generative technology* since it allows representing variability of design specifications at a higher abstraction level concisely and those specifications can be supported by automatic tools.

Here we speak about the *heterogeneous meta-programming techniques*, which use two different languages in the same specification: a *meta-language* for representing higher-level manipulations and *domain (or target) language* for representing domain program instances. As typically LO instances are represented in XML format they are actually domain programs. But meta-programming techniques can be used to the plain text and pictures which can be viewed as non-executables domain programs (though showing them holds execution too since they are interpreted, e.g., by the Internet Explorer).

5.2. Developing generative specification

The central point in the development of the generative specification (aka meta-program or simply GLO) using the meta-programming approach are the following phases: 1) introduction of a model for representing the GLO (model 1); 2) use of a LO model for representing LO instances (model 2); 3) identification of commonality-variability in the explicit form for the given family of related LO instances (model 3); 4) selection of a meta-language; 5) transformation of models 2 and 3 into meta-constructs of the selected language in compliance with model 1; 6) testing and validation of GLO specifications; and 7) incorporation of a GLO into a eLearning environment. As quality aspects are crucial, we admit that tasks in phase 6 and 7 should be considered separately.

Model 1 is represented in Figure 2. It contains meta-interface and meta-body. The latter consists of generic interface and generic body. The meta-body is some generalization of model 2 (see Figure 1), whereas the knowledge-based interface and the knowledge-based body are generalized by introducing meta-operations with parameters obtained as a result of construction of variability model 3 (see Table 1). The values of the parameters to implement the gene-

ralization are described in the meta-interface of the specification.

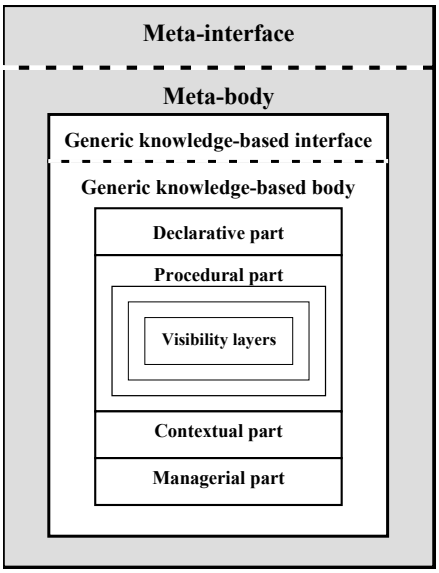

Figure 2. Structure of GLO model

We have selected our own meta-language Open PROMOL (see our web site *www.soften.ktu.lt/~damarobe/promol*) because: (1) it contains a wide spectrum of commands for the manipulation with text; (2) it is based on the use of functional approach and its functions are close to the syntax of the most known programming languages; (3) we have a long experience in using it in teaching and learning; (4) it has international recognition and (5) it has clear (human readable and computer readable) meta-interface. The latter is the most essential requirement for the LO domain because the LO user is usually not a programmer.

The models' transformations follow the rule: variability parameters are represented as meta-parameters at a higher abstraction level in the meta-interface, while the commonality-variability relationships are coded at a lower-level within the meta-body. This is explained in Section 5.3 by examples.

```
@- this is the beginning of meta-interface
$
              "Identify function from the list:"              {AND, OR}       f:= OR;
              "Identify number of inputs:"                    {2..10}         n:= 3;
[f eq {AND}] "Select symbolic representation of AND function" {and, ∩, &, .} rep:= and;
[f eq {OR}]  "Select symbolic representation of OR function"  {or, ∪, !, +}  rep:= or;
$
@- this is the beginning of meta-body

OUTPUT KNOWLEDGE IS Definition of "@sub[f]" Boolean function
in the narrative and truth table forms with an example:

@case[##f+1,{
Logical conjunction (usual symbol @sub[f]) is the function that results in a value of
TRUE if all of its operands are TRUE, otherwise a value of FALSE.
},{
Logical disjunction (usual symbol @sub[f]) is the function that results in true just
whenever some of its arguments are true.
}]

The truth table of p @sub[f] q is as follows:
       p       q       p @sub[f] q
       FALSE   FALSE   @case[##f+1,{FALSE},{FALSE}]
       FALSE   TRUE    @case[##f+1,{FALSE},{TRUE}]
       TRUE    FALSE   @case[##f+1,{FALSE},{TRUE}]
       TRUE    TRUE    @case[##f+1,{TRUE},{TRUE}]

This is an example of "@sub[f]" Boolean function with @sub[n] inputs:
             Y = @gen[n, { @sub[rep] }, {X}, 1];
```

(a)

```
OUTPUT KNOWLEDGE IS Definition of "OR" Boolean function
in the narrative and truth table forms with an example:

Logical disjunction (usual symbol OR) is the function that results in true just whenever
some of its arguments are true.

The truth table of p OR q is as follows:
       p       q       p OR q
       FALSE   FALSE   FALSE
       FALSE   TRUE    TRUE
       TRUE    FALSE   TRUE
       TRUE    TRUE    TRUE

This is an example of "OR" Boolean function with 3 inputs:
Y = X1 or X2 or X3;
```

(b)

Figure 3. GLO example (a) and instance (b) generated according to values (*f:*= OR*; n:=3; rep:*=or;)

5.3. Case study: GLO for learning homogeneous Boolean functions

The use of the GLO may have several formats (e.g., text, pdf, XML-based, others internet-based). In our case study we use the textual format (Figure 3) and HTML-based (Figure 4).

In Figure 3 (a) and (b), the GLO specification and its generated instance are given, respectively. Note that the specification represents a simplified version of the implementation of the LO and variability models discussed in Sections 3 and 4. Furthermore, meta-body fragments are shown only (compare the description in 3.6 and the description in Figure 3(b)). The specification of GLO (Figure 3(a)) is presented as a meta-program in Open PROMOL meta-language (its meta-functions are in bold). The meta-interface of the GLO presents a number of options for a teacher/learner. He or she can select a Boolean function, its notation, the number of inputs for the demonstration of the Boolean function.

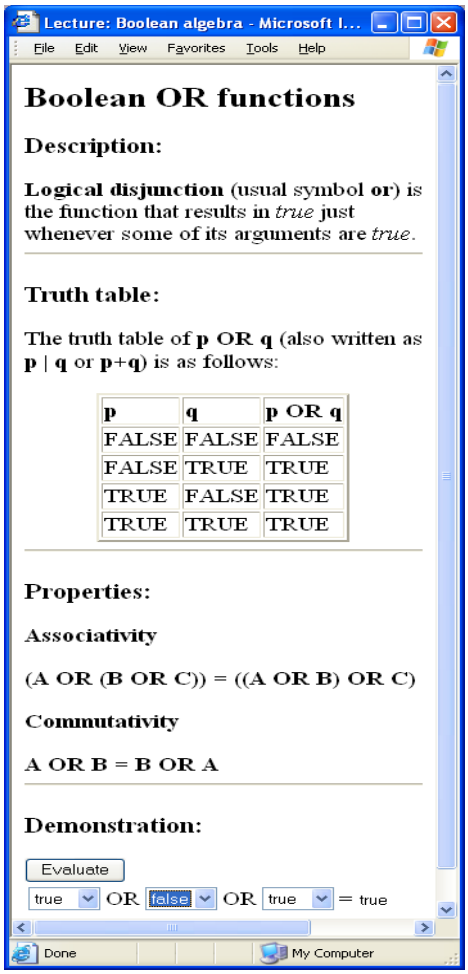

Figure 4. An example of specific Boolean function LO instance

Using the specified interface options, the meta-language processor generates a LO implemented in HTML+Javascript, which can be distributed over the internet. The HTML part of the LO is used for presentation of the description and truth table of the selected Boolean equation, while Javascript is used for demonstration of the Boolean equation. An example of the generated specific Boolean function LO as seen via the internet browser is given in Figure 4.

6. Discussion and evaluation

6.1. Summary and contribution

Reusability is a fundamental feature of LOs. Though reusability aspects, such as the component view to LOs or the role of granularity for reusable components and models, were at the focus of researches for long time, the efforts to enhance these aspects towards generative reuse are expending only now. Examples of this shift are the introduction innovative models that support adaptation of LOs (e.g., glass-box reuse model, aspect-oriented LOs, adaptive LOs or LOs based on object programming concepts) and the concept of generative LOs. In this paper, we have shown of how the dimension of generative reuse of LOs can be extended further. Firstly, we have suggested a novel knowledge-based LO model to represent component instances. The model is based on factoring and aggregating knowledge units within a LO and consists of two basic parts: *interface* and *functionality*. Interface is for the explicit description of knowledge communication in both directions: *to* a LO and *from* the LO. Functionality is for describing knowledge representation and managing. The model contributes to better compositionality, understandability, extensibility, reusability and can be further generalized easily to support the personalized content delivery and automatic generation. Secondly, we have extended the known concept of generative LOs by connecting commonality-variability analysis in the domain of related LOs with heterogeneous meta-programming techniques for generating LO instances on demand from the generic LO specifications.

6.2. Advantages

The advantages of knowledge-based generative LOs are: 1) better quality due to GLOs could be developed and tested by experts; 2) increase in productivity due to LO instances are generated from the generic LO specification on demand automatically; 3) managing of related LO instances within a CMS is simpler due to they are generated from a single specification and then distributed to learners in a variety of instances automatically; 4) flexibility for changes and adaptations due to the generative technology selected; 5) even the same material within educational systems can be easily tailored to both students' and teachers' individual needs and presented as a GLO; 6) as the approach describes spaces for relative LOs, the GLOs may be a part of the meta-design environments of eLearning systems (in the sense of the G. Fisher's *et al*. meta-design concept).

6.3. Disadvantages

The disadvantages of the approach are: 1) higher costs due to the need of additional efforts for analysis; 2) the need of the additional tool support for the generative technology; 3) more complicated testing; 4) the need for extensive experimentation in order to achieve a higher maturity; 5) generative approach requires more precise explicit LO models.

6.4. Problems for further work

As the related learning content may have a large variability dimension, a GLO can be viewed as a concise specification of the whole family of related LO instances. But it is not quite clear of how large or small GLOs should be and what the scope of GLOs variability is optimal. The mechanism of explicit integration of knowledge units within a knowledge-based LO instance is to be further extended, as well as the external composition of smaller LOs into lager ones. To enhance reusability, GLOs and their supporting environment are to be integrated into eLearning systems (e.g., Content Management Systems). These problems require further research activities.

7. Conclusions

The explicit representation of input-output knowledge within the proposed knowledge-based LO model gives (for course designers and teachers) a basis for the systematic construction of courses from LO instances since the model describes interaction explicitly. Learners are better guided for self-learning due to the knowledge is represented at two abstraction levels: interface and functionality. Commonality-variability analysis brings a solid background to construct a systemic approach to build generative LOs easily tailored to both learners' and teachers' individual needs. The heterogeneous meta-programming technology suits well for representing variability explicitly and implementing generative LOs. What we suggest, in order to enhance the generative reuse dimension in the LO domain, is 1) to incorporate commonality-variability analysis in instructional design theories applied by the instructional designer in the development of GLOs; and 2) to integrate the LO instances generation process into the content delivery process.

References

[1] **V. Rossano, M. Joy, T. Roselli, E. Sutinen.** A Taxonomy for Definitions and Applications of LOs: A Meta-analysis of ICALT papers. *Educational Technology & Society*, 8 (4), 2005, 148-160.

[2] **P. Fournier-Viger, A. Mayers, M. Najjar, R. Nkambou.** A Cognitive and Logic Based Model for Building Glass-Box Learning Objects. *Interdisciplinary Journal of Knowledge and Learning Objects*, *Vol*. 2, 2006, 77-94.

[3] **V. Ponkratius.** Aspect-Oriented Learning Objects. *Proc. of the IASTED International Conference on WEB-BASED EDUCATION, February, Grindelwald, Switzerland*, 2005, 21-23.

[4] **A. Berlanga, F.J. Garcia.** A Proposal to Define Adaptive Learning Designs. *Proceedings of Workshop on Applications of Semantic Web Technologies for EducationalAdaptive Hypermedia* (SW-EL 2004) *in* AH 2004, 23 *August*, 2004, *Eindhoven, The Netherlands. TUE Computer Science-Reports* 04-19 AH2004: *Workshop Proceedings Part* II.

[5] **P. Mohan, Ch. Brooks.** Engineering a Future for Web-based Learning Objects. *Proc. Of Web Engineering, International Conference, Oviedo, Spain, July* 14-18, 2003. *Lecture Notes in Computer Science* 2722 *Springer* 2003, 120-123.

[6] **T. Boyle, D. Leeder, H. Chase.** To boldly GLO – Towards the Next Generation of Learning Objects. *World Conference on eLearning in Corporate, Government, Healthcare and Higher Education*, *Washington USA*, *Nov*. 2004.

[7] **R. Morales, D. Leeder, T. Boyle.** A Case in the Design of Generative Learning Objects (GLO): Applied Statistical Methods GLOs. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, Montreal, Canada, June* 27-*July* 2, 2005, 302-310.

[8] **M.D. Merrill.** Instructional Transaction Theory (ITT): Instructional Design Based on Knowledge Objects. *C.M. Reigeluth (Ed.), Instructional-Design Theories and Models: A New Paradigm of Instructional Theory*, *Mahwah, NJ: Lawrence Erlbaum Associates*, 2000.

[9] **D.A. Wiley.** Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy. *D.A. Willey (Ed.): The instructional Use of Learning Objects*, 2002.

[10] **G.H.J. Redeker.** An Educational Taxonomy for Learning Objects. *Proceedings of the* 3*rd International Conference on Advanced Learning Technologies* (ICALT'03), *IEEE*, 2003.

[11] **J. Coplien, D. Hoffman, D. Weiss.** Commonality and Variability in Software Engineering. *IEEE Software*, *November/December*, 1998, 37-45.

[12] **J. Sametinger.** Software Engineering with Reusable Components. *Springer*, 1997.

[13] **T. Sheard.** Accomplishments and Research Challenges in Meta-Programming. *Lecture Notes in Computer Science*, *Vol*.2196, 2001, 2-44.

[14] **T.H. Davenport.** Information Ecology. *Oxford University Press*, 1997.

Received January 2007.