Auto-Generative Learning Objects for IT Disciplines

Ciprian-Bogdan Chirila[1]

(1) Department of Computer and Information Technology,
University Politehnica Timişoara, ROMANIA
E-mail: chirila[at]cs.upt.ro

Abstract

The 21-st century skills and key competences include domains like: mathematical literacy, basic competences in science and technology, digital competences. Other important dimension considered by the european commission is the empowerment of european citizens through the digital literacy. The european IT development trend for a digital society, digital economy and IT based research and inovation requires a significant number of specialists which are hard to train with current academic resources and capacities. Learning objects LOs together with their hierarchical structures tend to be the bricks for building knowledge managed by Learning Management Systems (LMSs). Auto-generative learning objects (AGLOs) are innovative learning technologies based on reusable templates instantiated with data based on random number generators to favor the variability of learning object content. AGLOs tend to be an innovation that could support future virtual learning and long-life learning. AGLOs advantages are twofold: i) they offer automatic variability and training facilities for students and ii) provide an easy way, tool assisted for the creation of learning objects. We consider that AGLOs generative feedback is a powerful feature that helps the student to better understand the content eliminating eventual doubts. We will show that AGLOs can support several IT disciplines like: Data Structures, Functional Programming, Compiling Techniques, Operating Systems with several limitations. In order to simplify creation of AGLOs specialized libraries are developed in this sense thus several levels of intervention on content is previewed: i) easy modifications - text decorations, notations; ii) structure modifications - use of library functions; iii) library modifications - addition of new functionalities.

Keywords: learning objects, auto-gebnerative learning objects, IT disciplines, generative models

1 Introduction

Future key competences and skills include domains like: mathematical literacy, science and techology, digital competences. Big efforts are consumed in the empowerment of citizens through digital literacy because: commerce tend more and more to be digital, state administration uses software online applications to offer services to its citizens, health keeping medical devices are electronic etc. To sustain such goals there is a need for IT specialists which are trained in universities, specialized schools, companies etc.

Virtual learning is an important instrument in forming IT specialists. I this context Learning Objects (LO) are the bricks for this construction. Over time LOs were specialized as generative learning objects (GLOs) which are reusable patterns instantiated with variable content and auto-generative learning objects (AGLOs) where the generation is automatic depending on random numbers.
In this paper we will show how the generative open response exercise ideas for several IT disciplines developed in (Chirila, 2014 SOFA) can be implemented using AGLO models proposed in (Chirila, 2015 BRAIN).
The paper is structured as follows. In section 2 we present briefly the model of AGLOs. In section 3 we show how the generative exercises can be implemented using the AGLO model. Section 4 presented related works. Section 5 concludes and sets the future work.

2 The AGLO Metamodel in a Nutshell

The AGLO model was developed in several interations: (Chirila 2013, Chirila 2014 SOFA and Chirila 2015 BRAIN). The original goal was to develop a generative model allowing to express variable questions, answers and feedbacks for practical problems built as dialog games for primary and middle school students. The main idea of the model is to use expression instantiated variables based on random numbers. Thus, we can generate values to fit for a certain scenario, where the student can exercise his competences.
For example, in order to test the understanding of arrays memory representation we can choose randomly a starting address for the array and then to ask the memory address of several array elements located at several indexes. If the first attempt of the student fails then he can exercise again on a different data set, so its competence to become accurate. On the other hand a targeted feedback on the current scenario can help him understand better the learned concepts.
Structuraly an AGLO is formed out of:
i) name – the name of the AGLO;
ii) scenario – the section where the variables are defined based on random numbers;
iii) theory – a short theoretical part to be show to the student to better accomplis his task;
iv) question – the section where a question is formed out of static text and variable values;
v) answer – the section where the answer is read from the student in the form of variables in order to be assessed automatically;
vi) feedback – the section where a feedback text is composed out of static text and values to explain the student the essence of the exercise with concrete details.
From the technical point of view variables, denoted as symbols, are instantiated with JavaScript (ECMA 2011) compatible expressions, then these values are catenated with inline strings to form the sentences of the AGLOs. Values are read from the student input and stored in variables in order to be checked against correct computed answers so to give the student feedbacks and marks.

3 AGLO Models for IT Disciplines

In this section we will present the exercise ideas of (Chirila, 2014 SOFA) in the context of the AGLO model.

For the problem of applying searching and sorting algorithms we need to generate a random array and then to compute the algorithm steps using a library. These steps are easilly generated in our prototype in the demonstrative learning objects. Array lines can be easily compared and we can identify automatically any errors. In case of an error we can detect the line and the position and try to identify the comparison or move that was not understood by the student in order to explain it. The same idea of writing the algorithm steps applies to string searching algorithms. The student input facilities must be very accurate when strings are overposed in order to find a match. The solution in this case is an editor with equal width characters like "Courier New" or a custom editor to provide such an editing feature.

When a tree must be drawn having a parent indicator array available we need a tree drawing tool to allow the student to build it. The output model can be compared against the array to see if all the child-parent relations hold and that there are no extra such relations. The same strategy can be applied to build a binary tree out of an integer array using a tree editor. The verification stands in checking that the inorder elements of the tree are equal to the initial sorted array. Erasing the root twice will result in several, four in this case, tree possibilities since the removal can be made with replacement from the left or from the right subtree.

For the graph exercise where a connection matrix is given and a graph must be drawn, we can use the diagram editor to compare the models. The graph degree exercise is a simple question with an integer answer trivial to assess.

For depth-first and breadth-first graph searches the results are not singular so a verification function is necessary for the assessment.

For the minimum spanning tree of Prim algorithm the solutions are not unique when edges with equal values are available at a certain decision point. One valid choice is to make the generation as such to avoid duplicates, but we consider such cases are good to show to students for a better understanding of the concept. For Prim algorithm the steps of choosing the closest node from the unvisited set and to push it into the visited set is quite easy to analyze and assess. For Krushkal algorithm the things are much more complicated since there are two stages of the algorithm: i) sorting the edges; ii) inserting the edges and two data structures to maintain: i) the newly built tree; and ii) the node sets to prevent cycles. Maybe here we should check only the student selected edges in the main exercise and for the rest of the details to design new exercises.

In the context of list programing the first exercise proposed to use primitives like CAR, CDR to extract a symbol from a multilevel list can be tested by evaluating the query written by the student or by determining the solution by computation.

In order to exercise the ability of writing Lisp functions the randomly generated expression trees can be traversed in preorder for the assessment purposes.

In the context of compiler design discipline, for dynamically generated grammars we have two parts: lexical analyzer and syntactical analyzer. The lexical analysis automaton can be checked by ignoring state numbers which may be different and checking only the

geometry of the diagrams. In order to be able to fulfill such an exercise strict rules must be set when allowing the student to design its automaton. The code written upon the automaton, namely the lexical analyzer itself, is difficult to be checked in general. A possible solution is to predefine the working data structures like: current character, atom buffer, current state variable and to show a short example of the code the student will have to write. Assessing student written code correctness is still an open challenge in the context of evaluation since there are virtually infinite code constructions with the same semantics. One solution used in programming contest competitions is blackbox testing. This approach makes impossible the generation of any feedback. Formal checking models with abstractions have a potential in solving at least partially such problems.
In the syntactic analysis part the left factoring for ambiguities elimination can be easilly tested using symbolic computation but the syntactical analyzer code can be tested only if it is written following a strict template.
To exercise operating systems competencies the exercises proposed to write several command with different arguments in the same context is quite simple for the students. A better approach is to create a new environment of directories and files and to set different targets to be accomplished using commands. The answers can be easilly assessed by comparing the student command and arguments with the referenced ones according the the random context.

4 Related Works

The concept of learning object (Wiley, 2000) refers to reusable, transferable digital materials to be used for learning purposes. Standards were created in this sense e.g. Learning Oject Model (LOM) (IEEE LTSC, 2000).
The term of GLO was coined by (Boyle, 2003, Boyle 2006 and Jones et al., 2007). They implemented the idea of developing consumer LOs following a reusable pattern and filling in with content. GLOs are considered to be the second generation of LOs in this sense. GLOs adopt several principles from object-oriented programming. With our AGLOs we keep the pattern philosophy of GLOs but the generation is different.
The GLO approach of (Stuikys et al. 2007, 2008, 2009, 2013 and Damasevicius et al. 2008, 2009) use metaprogramming for the generation of content to be filled in the patterns. Our AGLO approach use metaprogramming for content generation but the main difference is that our generation is highly based on random numbers.
(MCQ, 2015) presents a model for creating generative learning objects very similar to AGLOs dedicated to mathematics and physics classes and they are integrated in Moodle (Moodle, 2015) learning management system. AGLOs rely on libraries that offer functionalities for IT disciplines.
Other uses of GLOs are presented in (Han, 2009) – where a Bloom taxonomy cognitive layers GLO is created equipped with pedagogical parameters and (Oldfield, 2008) – where a depreciation model from economy is taught using GLOs.

5 Conclusions and Future Work

In this paper we explained how the AGLO model helped by library extensions can cover several IT discipline exercises dealing with concepts like: data structures and algorithms,

functional programming, lexical and syntactical analyzers, operating system commands. As future work we intend to implement a full curricula of AGLOs for several disciplines in order to test them on students. Another research direction is in the area of creation of AGLOs, of creating tools for design and generation in order to keep content construction at a lower complexity level.

6 Acknoledgement

This paper is supported by the Sectoral Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the project number POSDRU/159/1.5/S/134378.".

7 References

Journal Articles:

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

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

Jones, R.; Boyle, T. (2007), Learning Object Patterns for Programming, Interdisciplinary Journal of Knowledge and Learning Objects, vol. 3.

Jung, H.; Park, C. (2012), Authoring Adaptive Hypermedia using Ontologies International Journal of Computers Communications & Control, ISSN 1841-9836, volume 7(2), pp. 285-301, June.

Stuikys, V.; Damasevicius, R. (2007), Towards Knowledge-Based Generative Learning Objects, Information Technology and Control, 36(2), ISSN 1392-124X.

Stuikys, V.; Damasevicius, R. (2008), Development of Generative Learning Objects Using Feature Diagrams and Generative Techniques, Journal of Informatics in Education, volume 7(2), pp. 277-288, Institute of Mathematics and Informatics, Vilnius, Lithuania.

Stuikys, V.; Brauklyte, I. (2009), Aggregating of Learning Object Units Derived from a Generative Learning Object, Journal of Informatics in Education, volume 8(2), pp. 295-314, Institute of Mathematics and Informatics, Vilnius, Lithuania.

Stuikys, V.; Burbaite, R.; Damasevicius, R. (2013), Teaching of Computer Science Topics Using Meta-Programming-Based GLOs and LEGO Robots, Journal of Informatics in Education, volume 12(1), pp. 125-142, Institute of Mathematics and Informatics, Vilnius, Lithuania.

Conference Proceedings:

Boyle, T. (2006), The design and development of second generation learning objects, Invited talk at Ed Media 2006, World Conference on Educational Multimedia, Hypermedia and Telecommunications, Orlando, Florida, June 28.

Chirila, C.B. (2013), A Dialog Based Game Component for a Competencies Based E-Learning Framework, In proceedings of SACI 2013 8-th IEEE International Symposium on Applied Computational Intelligence and Informatics, pp. 055-060, Timisoara, Romania, May.

Chirila, C.B. (2014), Educational Resources as Web Game Frameworks for Primary and Middle School Students, In proceedings of eLSE 2014 International Scientific Conference eLearning and Software Education, Bucharest, Romania, April.
Chirila, C.B. (2014), Generative Learning Object Assessment Items for a Set of Computer Science Disciplines, In proceedings of SOFA 2014 6-th International Workshop on Soft Computing Applications - Advances in Intelligent and Soft Computing, Springer Verlag, ISSN 1867-5662, Timisoara, Romania, July.
Damasevicius, R.; Stuikys, V. (2008), On the Technlogical Aspects of Generative Learning Object Development, Third International Conference on Informatics in Secondary Schools - Evolution and Perspectives (ISSEP 2008), pp. 337-348, Torun, Poland, July.
Damasevicius, R.; Stuikys, V. (2009), Specification and Generation of Learning Object Sequences for e-Learning Using Sequence Feature Diagrams and Metaprogramming Techniques, In proceedings of 2009 9-th International Conference on Advanced Learning Technologies, pp. 572-576, Riga, Latvia.
Han, P.; Kraemer, B.J. (2009), Generating Interactive Learning Objects from Configurable Samples, In proceedings of International Conference on Mobile, Hybrid and On-line Learning, pp. 1-6, Cancun, Mexico, February.
Oldfield, J.D. (2008), An implementation of the generative learning object model in accounting, In proceedings of Asciilite (Australasian Society for Computers in Learning in Tertiary Education), Melbourne, Australia.

Manuscripts And Working Papers (Unpublished Material):

Chirila, C.B.; Ciocarlie, H.; Stoicu-Tivadar, L. (2015), Generative Learning Objects Instantiated with Random Numbers Based Expressions, Broad Research in Artificial Intelligence and Neuroscience (BRAIN), October, (to appear).

Internet Sources:

ECMA International: Standard ECMA-262 ECMAScript Language Specification Edition 5.1 (2011), http://www.ecma-international.org/publications/standards/Ecma-262.htm (accessed in September 2015).
IEEE Learning Technology Standards Commitee (2000), LOM working draft v4.1, http://ltsc.ieee.org/doc/wg12/LOMv4.1.htm (accessed in September 2015).
Wiley, D. (2000), Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. D. Wiley (Ed.), The instructional use of learning objects: Online version. Retrieved January 05, 2015 from http://reusability.org/read/chapters/wiley.doc (accessed in September 2015).

Computer Programs:

Moodle Coordinate Question Plugin (2015), Tutorial and Documentation, https://code.google.com/p/moodle-coordinate-question.
Moodle Pty Ltd (2015), Moodle Open Source learning platform, http://www.moodle.org.