DYNAMIC DEVELOPMENT AND ASSEMBLY OF LEARNING OBJECTS IN A MATH LEARNING ENVIRONMENT

Stănică Justina Lavinia [1]
Crișan Daniela Alexandra[2]

Abstract:

The current research intended to define and implement a learning object framework, which has been applied in the development of an integrated learning environment for the mathematics field of study. This approach presents many advantages, since it guarantees, through components combination, the functional characteristics of a learning software, by enabling the teacher to control the lesson configuration and to adapt it to the classroom level, and by adjusting the instruction to the many possible approaches and multiple solutions to the problems.

Keywords: **e-Learning, learning object, learning environment, learning architecture**

1. Introduction

In the past decade, education theorists have described the way technology has changed and keeps changing education. No doubt, the technological development continues to provide new opportunities for training and offers viable alternatives for the traditional teaching / learning process.

However, qualitative e-Learning resources are still expensive to produce and training efficiency and investments refund are generally limited by several factors. A problem would be that the educational resources created within a learning platform usually cannot be transferred to be used in other training applications. From another point of view, teachers often want to reuse or modify the learning materials to suit the needs of their students, but most educational software do not allow this.

Trying to overcome these drawbacks, the recent researches have anticipated the way reusable learning objects will reduce the costs of e-Learning, while increasing its quality and accessibility. Learning objects are seen as the technology of the future for designing, developing and delivering the next generation of training software, due to the adaptability, reusability and scalability potential they provide.

Given the advantages pointed above, the present paper intended to find a methodology for applying this technology, and to design a flexible framework for assembling and reusing learning objects. The practical aspects of our research have resulted in implementing this technology in an educational platform, in order to increase its quality, reusability, and productivity. Therefore, an innovative software solution for authoring and learning

[1] Lecturer, PhD., Romanian-American University, Bucharest, email: fercalalavinia@yahoo.com
[2] Lecturer, PhD., Romanian-American University, Bucharest, email: dacrisan@yahoo.com

mathematics has been developed, which allows teachers to create and customize their own lessons by easily configuring and combining instructional components.

2. Learning Objects Technology

Even since 1999, the concept of learning objects has been widely used within the e-Learning communities. However, the terms *learning object* and *reusable learning object* are often used excessively, thereby reducing them to formal expressions. The lack of conceptual clarity is obvious in a multitude of definitions and uses of these terms.

This technology aims to facilitate the interchange of learning resources between courses, curricula, teachers, or even institutions. The main idea is that teachers should reuse the existing educational materials, instead of creating new ones every time they develop a course.

The problems concerning learning objects and correlated technologies begin with their own definition. From an operational point of view, these are small instructional pieces, used in an e-Learning system – which are created, stored, labeled, assembled or referred [KNOL03]. In a more practical approach, they can be considered as fragments or parts of a course that can vary in size and complexity from a simple chart to the whole course.

Although there are differences of opinion that mark the defining of learning objects [ALLE10], the practical implementation of this technology requires that the objects should have some common characteristics. Therefore, they must be:

- self-contained – an object must describe a single concept, so that if it was used in another context, it should remain perfectly intelligible and functional;
- independent – a learning unit should have minimal links to other resources, therefore it should not refer or use elements of other objects;
- reusable – an object can be used in various contexts, for multiple learning purposes, on different learning environments; the idea is to create a single time and distribute many times;
- of low granularity – small-sized objects provide flexibility; the main assumption is that the smaller an object is, the easier it can be reused in new learning situations;
- can be combined – small-sized components can be combined to create more complex units of learning; for objects with greater granularity, the combination is more difficult, because of the many types of integrated elements and the links between them;
- have potential for personalized learning – the existence of multiple possibilities for objects coupling facilitates the adaptation of the courses to the students' learning requirements;
- tagged with metadata – objects' labeling and description will allow an easy and accurate identification and searching.

However, the main benefit of learning objects is their reuse potential. The reusability facilitates the educational resources exchange between software developers, organizations, institutions, or teachers. Thanks to this interchange, the objects acquire value; without it, they remain simple digital components or web pages.

3. Learning Objects Granularity

Learning objects granularity is an important feature referring to the objects size. The components small dimension is a fundamental requirement to facilitate their reuse. The main hypothesis is that if an object is smaller, it is easier to reuse it in new training contexts [BALA08]. In these situations, the components should be combined with others to form more complex units of learning. Small-sized objects are more flexible, based on the fact that the same training material can be used repeatedly.

Starting from this idea, one solution adopted in the current approach consisted in designing learning objects of different levels of granularity. Therefore, three categories were defined: *fundamental objects*, *instructional components*, and *compound objects*.

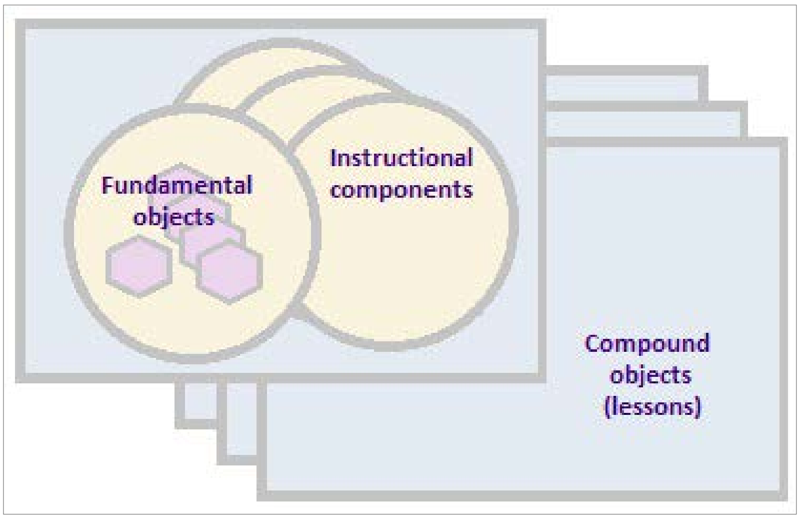


Figure 1. Learning objects granularity

The simplest object is a fundamental component, for example a text, a function, or a mathematical formula. They will underpin the development of instructional objects and will be used to manage their interaction. An instructional component gains value when it is included in a lesson. More components can be combined to create a lesson, and at the same time, a component can be used in different lessons. In order to ensure a longer training experience, the lessons are grouped in chapters according to the curriculum.

4. Fundamental Objects

These are basic components having a low granularity, which do not pursue a training objective. They were developed in order to manage the assembling of instructional components. In this category we have:

- ***LatexForm*** – this component was introduced with the purpose of ensuring the conversion of other objects into Latex format. Such a format provides a platform-independent way of writing mathematical equations, by using a textual language that facilitates the storage, interpretation, management, or compression of formulas. The mathematical functions and formulas associated to the instructional objects can then be stored, reused, or construed, so that to be compiled and evaluated by other components. The LatexForm objects are part of the TexEq components that can display the formulas in a visual equational form.

- ***Function*** – this is an object specific to the mathematics field of study. It is designed to handle one-variable mathematical functions, which are defined by their analytical expressions. Such Function objects are included in instructional components of type FctEval (performs the syntactic validation of the analytical form), FctGraph (draws the function graph), FctTable (tabulates the function values), FctGraphCmp (draws a comparative graphical representation of several functions), and FctTableCmp (tabulates the values of several functions). In addition, the equational visualization of the function formula is provided by TexEq components; this can be done due to the conversions to and from LatexForm type.

The Function class is designed to cover an important mathematical concept. Every function is implemented through several analytical expressions; also, an essential attribute of a function is its domain of definition. Thus, a Function object can be further decomposed into components of lower granularity, given that it incorporates other objects that implement the analytical expressions and their associated intervals. For each of its expressions, a Function object will contain one member of type ExprPol (to implement the analytical form) and one member of type Interval (to handle the associated interval).

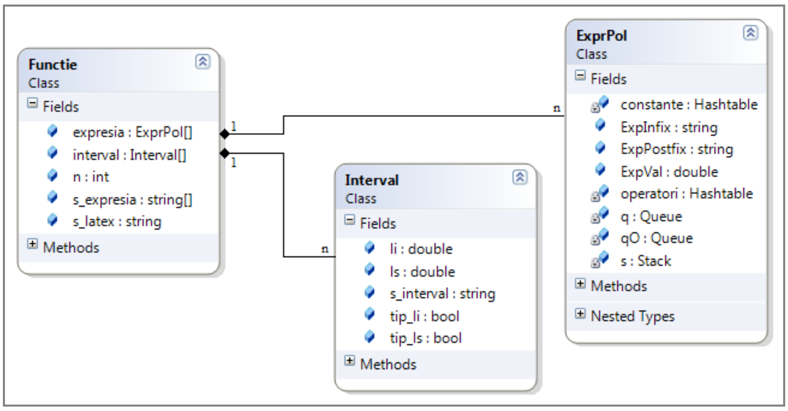


Figure 2. Function, ExprPol and Interval classes

In addition to compiling and evaluating mathematical expressions, the Function class offers the possibility of calculating the derivatives of real-valued functions. This operation is handled by the interaction between two FctEval objects: the source control will contain the original mathematical function, whereas its derivative will be calculated in the destination component.

The conversion between these two objects (LatexForm and Function), allows the user to choose editing the mathematical expressions in computational language or in Latex. Thus, if the author prefers to edit the function in Latex or to retrieve it from a TexEq object in this format, the associated Function object will be generated by the FunctionFromLatex method.

```
public static Function FunctionFromLatex(LatexForm LF)
{
    string latex = LF.latex_string;
    latex = latex.Replace('\\', '/');
    latex = latex.Replace("f(x)=", "");
    latex = latex.Replace("/left/{/begin{array}{", "");
    latex = latex.Replace("///end{array} /right.", "");
    latex = latex.Remove(0, latex.IndexOf('}') + 1);
    Function F = new Function();
    latex = F.ReplaceDinLatex(latex);
    latex = latex.Replace("//", ";");
    string[] tokens = latex.Split(';', (char)(StringSplitOptions.RemoveEmptyEntries));
    F.n = tokens.Count() / 2;
    for (int i = 0; i < F.n; i++)
    {
        F.s_expresia[i] = tokens[2 * i].Trim();
        F.interval[i] = Interval.IntervalDinLatex(tokens[2 * i + 1].Trim());
        F.expresia[i] = new ExprPol(F.s_expresia[i]);
    }
    return F;
}
```

The mathematical functions associated with instructional objects will be stored in the database in Latex format. In addition, the TexEq components work with the same textual format and no longer contain a Function object. Thus, storing the mathematical functions and managing the interactions of TexEq objects with other controls, are facilitated by the reverse conversion to Latex format, which is performed by the FunctionToLatex method.

```
public LatexForm FunctionToLatex()
{
    string latex = "f(x)=/left/{/begin{array}{";
    string align = "";
    for (int i = 0; i < n; i++) align += 'l';
    latex += align + "}";
    for (int i = 0; i < n; i++)
    {
        string expresia = s_expresia[i];
        expresia = expresia.Replace("/", " / ");
        latex += expresia;
        latex += "; x" + (interval[i].s_interval.Contains(",") == true ? " /in " : " = ");
        latex += interval[i].s_interval;
        latex += "// ";
    }
    latex += " /end{array} /right.";
    latex = ReplaceCatreLatex(latex);
    latex = latex.Replace('/', '\\');
    latex = latex.Replace(" \\ ", "/");
```

```
        s_latex = latex;
        LatexForm LF = new LatexForm(latex);
        return LF;
    }
```

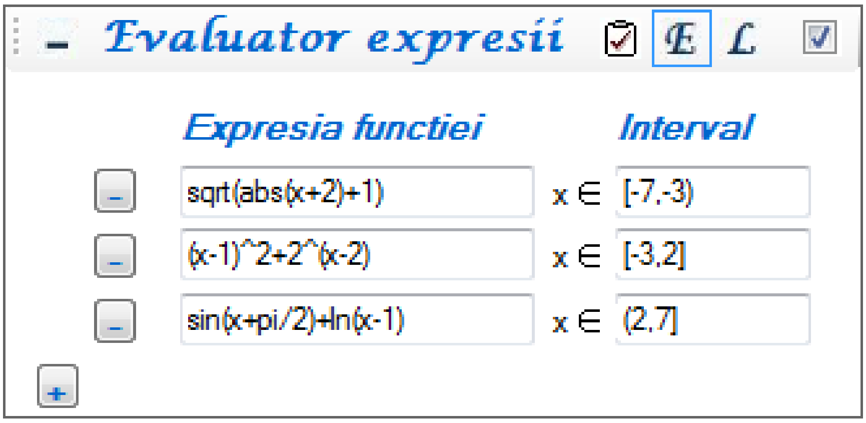
5. Instructional Components

The instructional components were defined with the purpose of covering some important mathematical notions related to the concept of function. They can be used or reused independently, or can be combined or assembled in order to effectively develop new mathematics lessons.

The behavior of these instructional components is determined by their assembling and coupling possibilities. The mathematical functions on which they operate are given through fundamental objects. Therefore, each instructional component includes one or more fundamental objects of type Function or LatexForm, that also have the role to control the way they interact.

- ***FctEval*** (function evaluator) – this component was introduced with the purpose of handling real-valued functions. The functions are given through one or more analytical expressions and their associated intervals of definition. The control performs the syntactic validation, analysis and evaluation of the function expressions, as well as a validation of the definition intervals.

The user can input the function in both supported formats (Latex or usual language), due to the existing conversions between fundamental objects (from Function type objects to LatexForm, and vice versa).



Expresia functiei		Interval
sqrt(abs(x+2)+1)	x ∈	[-7,-3)
(x-1)^2+2^(x-2)	x ∈	[-3,2]
sin(x+pi/2)+ln(x-1)	x ∈	(2,7]

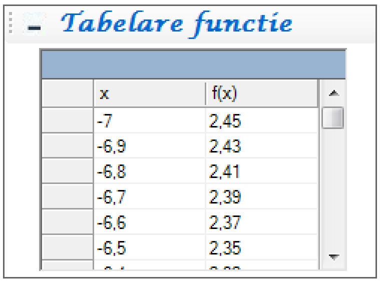
Figure 3. FctEval component

When the user requests the evaluation of the mathematical function, a Function type object is automatically generated. The Function object is associated with the component and will be used for the subsequent interactions with other controls.

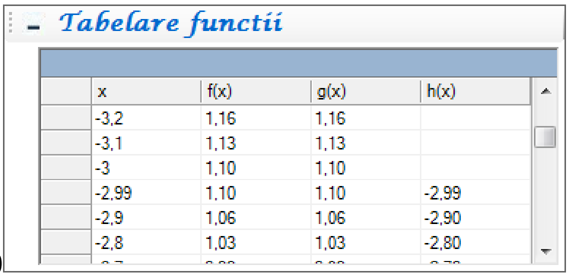
The buttons attached to the control enable the management of all the function expressions. When adding or deleting an expression, the associated Function object is automatically recalculated. This is done by two methods belonging to the Function class.

The process of evaluating the function consists of four tasks: syntactic analyzing the function expressions, validating the intervals associated with the expressions, rearranging the expressions in the order determined by their intervals, and calculating the function values in the definition domain.

- ***FctTable*** and ***FctTableCmp*** (tables of function values) – these two types of components calculate the values of one or more functions on their domain of definition. These two objects are alike, the difference between them being that the FctTable component works with one function, whereas the FctTableCmp control operates with more functions.

(a) 

x	f(x)
-7	2,45
-6,9	2,43
-6,8	2,41
-6,7	2,39
-6,6	2,37
-6,5	2,35

(b) 

x	f(x)	g(x)	h(x)
-3,2	1,16	1,16	
-3,1	1,13	1,13	
-3	1,10	1,10	
-2,99	1,10	1,10	-2,99
-2,9	1,06	1,06	-2,90
-2,8	1,03	1,03	-2,80

Figure 4. FctTable (a) and FctTableCmp (b) components

Like other instructional components, these controls also contain one or more Function class objects, which are copied from a FctEval control, as a consequence of their interaction. Thus, this operation will cause calculating the functions values, generating a DataSet containing these values, and loading it in a DataGrid object.

Furthermore, a one-directional interaction between FctTable and FctTableCmp has been defined, which determines the addition of a new function to the second object.

- ***FctGraph*** (graph of a function) – this object is designed to draw a function graph over the definition range. The function to be plotted will be taken from FctEval or FctTable components. Once again, the Function class object associated with the component is obtained from the similar object belonging to the control the current component interacted with. After the function is evaluated, the Graphics module will perform the tasks of calculating the axes and drawing the graph.

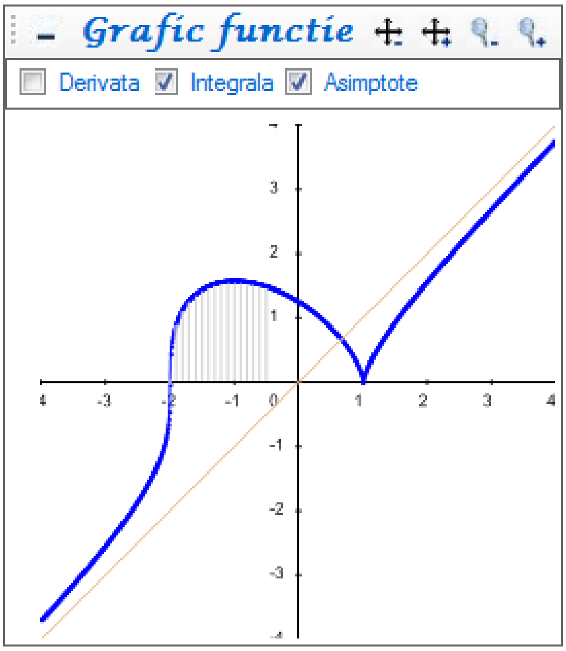


Figure 5. FctGraph component

This component can also be used to illustrate some other concepts related to the graphical representation of functions, namely:

- the derivative of a function at a point – this is the slope of the tangent line to the graph of the function at that point;
- the definite integral over an interval – its value is calculated as being the area of the region bounded by the graph of the function and the x-axis, over the given closed interval;
- the asymptotes of the function – the control facilitates the drawing of the horizontal, vertical or oblique asymptotes of the given function.

- ***FctGraphCmp*** (graph of multiple functions) – this component supports graphing more functions together. The functions to be graphed can be copied from a FctTableCmp control or can be added through repeated interactions with FctEval, FctTable, or FctGraph components.

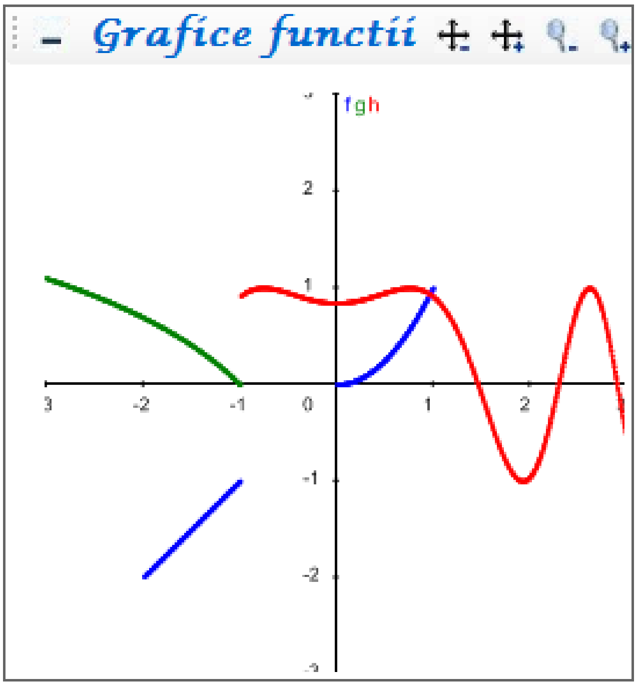
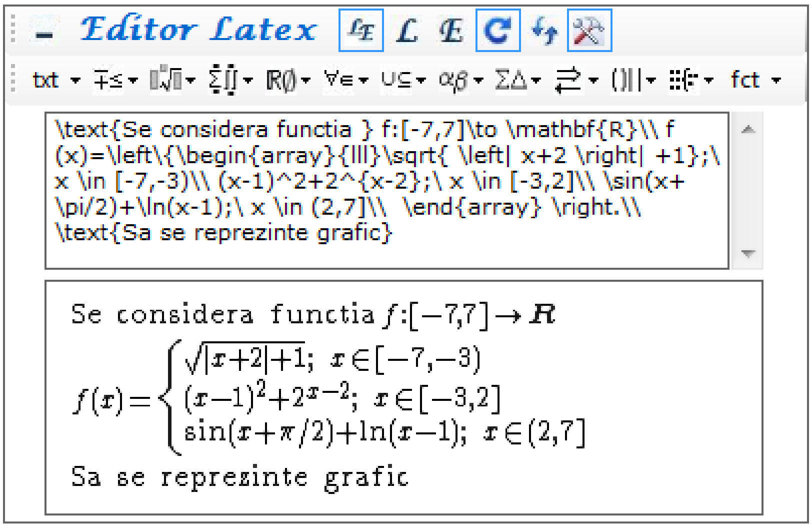


Figure 6. FctGraphCmp component

The mathematical functions that are tabulated or plotted by the FctGraphCmp and FctTableCmp components can be handled using the context menu associated with the control. Thus, the functions will be converted and displayed in Latex format and the user can choose to delete some of them. This will lead to updating the list of functions included in the component, and consequently recalculating the axes and redrawing the graph.

The toolbar integrated in the component contains buttons for resizing and scaling the graph. Resizing the object is a simple operation involving resizing the panel where the graph is drawn, whereas scaling causes the recalculation of the axes and redrawing the functions graphs according to the new axes.

- ***TexEq*** (Latex equation editor) – this component has the role of an equation editor, but it can also act as a simple text editor. Thus, the mathematical expressions are input using Latex syntax, which is then translated and displayed in visual equational form.



```
\text{Se considera functia } f:[-7,7]\to \mathbf{R}\\ f
(x)=\left\{\begin{array}{lll}\sqrt{ \left| x+2 \right| +1};\
x \in [-7,-3)\\ (x-1)^2+2^{x-2};\ x \in [-3,2]\\ \sin(x+
\pi/2)+\ln(x-1);\ x \in (2,7]\\  \end{array} \right.\\
\text{Sa se reprezinte grafic}
```

$$\text{Se considera functia } f:[-7,7]\to \mathbf{R}\\ f(x)=\left\{\begin{array}{lll}\sqrt{ \left| x+2 \right| +1};\ x \in [-7,-3)\\ (x-1)^2+2^{x-2};\ x \in [-3,2]\\ \sin(x+\pi/2)+\ln(x-1);\ x \in (2,7]\end{array} \right.\\ \text{Sa se reprezinte grafic}$$

Figure 7. TexEq component

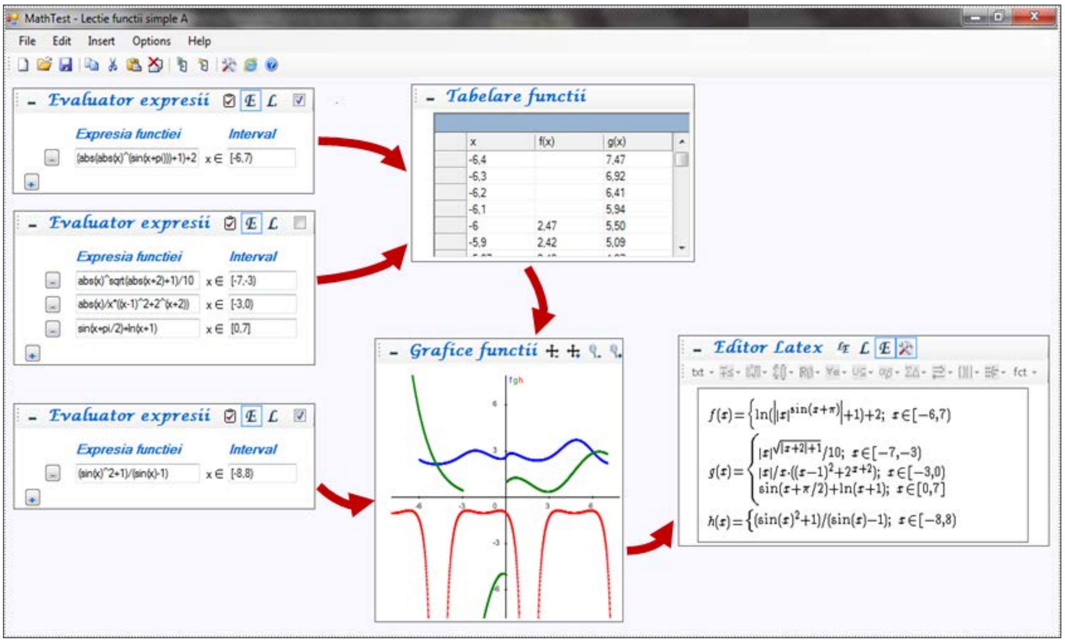
This type of object can also be used to visualize the functions associated with other instructional components. Even if TexEq controls no longer contain a Function object, they can interact with other objects due to the existing conversions to and from Latex format.

6. Compound Objects

Compound objects are complex objects that are created by combining more instructional components. The learning context is provided when the mathematics lessons are developed, as the pedagogical aspect is handled through the assembling operation.

Basically, a lesson consists of several interrelated instructional components. The software allows decomposing the lesson into elementary objects, which can be used independently or can be reassembled according to the teacher’s choice. Each lesson and the objects that compose it can be stored in the database, so that to allow their subsequent use. Consequently, the author might reuse an existing lesson or can reconfigure it by adding, removing, or modifying its components. The individual objects can also be reused separately of the lesson they belong to, in order to be used for creating new lessons. This feature facilitates the development of new learning content and more complex lessons.

Through simple drag-and-drop operations, the teacher is able to author his own mathematics lessons by assembling the components at his disposal. The objects can be coupled, configured, moved, and deleted, without affecting the remaining content. The coupling possibilities are numerous being determined, on one hand, by the instructional components variety and, on the other hand, by the need to illustrate the correlations between the mathematical concepts they implement.


MathTest - Lectie functii simple A

Evaluator expresii

Expresia functiei		Interval
(abs(abs(x)^(sin(x+pi)))+1)+2	x ∈	[-6,7)

Evaluator expresii

Expresia functiei		Interval
abs(x)^sqrt(abs(x+2)+1)/10	x ∈	[-7,-3)
abs(x)/x*((x-1)^2+2^(x+2))	x ∈	[-3,0)
sin(x+pi/2)+ln(x+1)	x ∈	[0,7]

Evaluator expresii

Expresia functiei		Interval
(sin(x)^2+1)/(sin(x)-1)	x ∈	[-8,8)

Tabelare functii

x	f(x)	g(x)
-6,4		7,47
-6,3		6,92
-6,2		6,41
-6,1		5,94
-6	2,47	5,50
-5,9	2,42	5,09

Editor Latex

$$f(x)=\begin{cases}\ln(\left||x|^{\sin(x+\pi)}\right|+1)+2; & x\in[-6,7)\end{cases}$$

$$g(x)=\begin{cases}|x|^{\sqrt{|x+2|+1}}/10; & x\in[-7,-3)\\ |x|/x\cdot((x-1)^2+2^{x+2}); & x\in[-3,0)\\ \sin(x+\pi/2)+\ln(x+1); & x\in[0,7]\end{cases}$$

$$h(x)=\begin{cases}(\sin(x)^2+1)/(\sin(x)-1); & x\in[-8,8)\end{cases}$$

Figure 8. Authoring a lesson by combining objects

The interactions between instructional components are handled through mediator objects that act as intermediary between interdependent components. This way of managing interactions facilitates components decoupling and ensures their independence. All components contain one or more such fundamental objects (Function or LatexForm), which, in addition to the role of mediating interactions, facilitate their copying and storing.

Interactions are implemented through simple drag-and-drop operations. The ease of combining the learning components, allows the teachers to effortlessly create and configure their own lessons.

7. Conclusion

A learning object approach has many benefits for creating educational resources and also for developing training platforms. The main reason for using this technology is given by its potential of reusing and customizing the learning materials. However, creating a lesson by combining independent low-granularity objects is problematic, given that the software must meet certain qualitative training requirements.

Therefore, the instructional components were designed in order to cover some important mathematical notions, while their interactions were defined considering the relations between these concepts. The coupling possibilities provide the functionalities needed for a math learning environment, facilitating the presentation of several field specific concepts, such as tabulating a function values or drawing a function graph. By using this platform, the teacher can combine the available items in order to create new lessons or to adapt the existing ones.

Given that recent years were marked by considerable efforts to promote computer-assisted learning, and most students prefer the information provided on the Internet, such an e-Learning solution meets these options.

8. Bibliography

1. [ALLE10] Allen C.A., Mugisa E.K., *Improving Learning Object Reuse Through OOD: A Theory of Learning Objects*, Journal of Object Technology, vol. 9, no. 6, pp. 51–75, ISSN 1660-1769, 2010, [http://www.jot.fm/issues/issue_2010_11/article3.pdf]
2. [BALA08] Balatsoukas P., Morris A., O'Brien A., *Learning Objects Update: Review and Critical Approach to Content Aggregation*, Educational Technology & Society, vol. 11(2), pp. 119-130, ISSN 1436-4522, ISSN 1176-3647, 2008, [http://www.ifets.info/journals/11_2/11.pdf]
3. [FRAN08] Francis D.E., Murphy E. *Instructional designers' conceptualisations of learning objects*. Australasian Journal of Educational Technology, vol. 24(5), pp. 475-486, ISSN 1449-5554, ISSN 1449-3098, 2008, [http://www.ascilite.org.au/ajet/ajet24/francis.html]
4. [KNOL03] Knolmayer G.F., *Decision support models for composing and navigating through e-learning objects*, Proceedings of the 36th Annual Hawaii International Conference on Systems Sciences, pg. 10, ISBN 0-7695-1874-5, 2003
5. [POLS03] Polsani P.R., *Use and Abuse of Reusable Learning Objects*, Journal of Digital Information, vol. 3(4), article 164, pg. 10, 2003, ISSN 1368-7506, [http://journals.tdl.org/jodi/article/view/89/88]
6. [SMEU08] Smeureanu I., Dârdală M., Reveiu A. *Component Based Framework for Authoring and Multimedia Training in Mathematics*. Proceedings of World Academy of Science, Engineering and Technology, vol. 29, pp. 230-234, ISSN 1307-6884, 2008