

# **MODELMAKER, A MULTIDISCIPLINARY WEB APPLICATION TO BUILD QUESTION GENERATOR MODELS FROM BASIC TO HIGHER EDUCATION**

**Jorge Camejo<sup>1</sup>, Alexandre Silva<sup>2</sup>, Luis Descalço<sup>1-2</sup>, Paula Oliveira<sup>1-2</sup>**

<sup>1</sup>*PmatE, University of Aveiro, Portugal*

<sup>2</sup>*Department of Mathematics, University of Aveiro. Portugal*

## **Abstract**

PmatE (Mathematics and Education Project) is a Research and Development project started in 1989 at the University of Aveiro, Portugal. For 27 years, PmatE has maintained the mission of applying technologies and developing content and events to foster school success and scientific culture. PmatE provides a large repository of learning objects, with particular emphasis to Question Generator Models (QGM) or simply Models. A QGM is an object for generating questions targeting specific scientific and pedagogical-didactic objectives. Each QGM generates thousands of different questions, thus enabling the exposure of students to the same core problems, but with different completions, preventing cheating in exams. The QGMs are the basis of Portuguese National Science Competitions (NSC), a three day yearly event with about ten thousand participants, and are widely used by schools nationwide, at various levels of education (from basic to higher education), for tests and diagnostic exams in several areas (e.g., Mathematics, Physics, Biology, Portuguese, Financial Education, Geosciences and Chemistry). Until September 2015, each QGM created was written in a LaTeX template, as an intermediate specification of the Model, and later implemented by dedicated programmers from PmatE, thus making the Model development and later corrections a tedious, lengthy and time-consuming task. This work presents PmatE *ModelMaker* solution, which enables professors with neither a coding background nor latex knowledge, from basic school to higher education from all areas mentioned above, to create and share QGMs through a Web application. *ModelMaker* keeps the core concepts of QGM such as “boxes” and “variables”, in order to guarantee the random screens concretization, and incorporates new functionalities and advantages (e.g., autonomy, model versioning, storage of instantiated models, and a management optimization of PmatE Subject Classification). Application *ModelMaker*, developed in ASP.NET, with SQL databases and Javascript frameworks, has proven to be a successful tool for saving time and giving total autonomy to QGM authors in the creation of the last 134 models in several areas of knowledge, as evidenced by the usage statistics extracted during the last eight months since the application has been available. This improvement in the QGMs development methodology is an important mark in the history of PmatE.

Keywords: QGM, Model, Education, Education Software, PmatE.

## **1 INTRODUCTION**

In the recent years, the researchers and professors have been witnessing an increasing interest in leveraging the recent advances of the Information and Communication Technologies (ICT) for providing better educational services. These have been evidenced by the emergence of several e-learning technologies targeting different areas. For example: (i) *Easygenerator* provides cloud-based eLearning authoring software, enables instructional designers and subject matter experts to create rapidly courses that have learning impact [1]. (ii) *SmartBuilder* is the award-winning course authoring tool that enables you to create rich Flash e-learning with an easy-to-use interface [2]. (iii) The *Multimedia Learning Object Authoring Tool* enables content experts to combine video, audio, images and texts easily into one synchronized learning object [3]. However, regardless of these platforms being very useful for creating courses and libraries of educational contents, they are not parameterized and therefore the randomness of contents is limited.

In this context, PmatE, a Research and Development project started in 1989 at the University of Aveiro [4], has maintained the mission of applying technologies and developing content and events to foster school success and scientific culture. Particularly, within the scope of PmatE, a novel approach for parameterized contents has been developed.

PmatE provides a large repository of learning objects, with particular emphasis in Question Generator Models (QGM) or simply Models. A QGM is an object for generating questions targeting specific scientific and pedagogical-didactic objectives [5]. Each QGM generates thousands of different questions, thus enabling the exposure of students to the same core problems, but with different completions, preventing cheating in exams.

The QGMs are the basis of Portugal National Science Competitions (NSC), a three day yearly event with about ten thousand participants, are also used in diagnostic tests [6, 7], and are widely used by schools nationwide, at various levels of education (from basic to higher education), for test and diagnostic exams in several areas (e.g., Mathematics, Physics, Biology, Portuguese, Financial Education, Geosciences and Chemistry). Until September 2015, each QGM created was written in a LaTeX template, as an intermediate specification of the Model, and later implemented by dedicated programmers from PmatE, thus making the Model development and later corrections a tedious, lengthy and time-consuming task.

This work presents PmatE *ModelMaker* solution which enables professors with neither a coding background nor LaTeX knowledge, from basic school to higher education, to build MGQ and make them available immediately to students, what was becoming imperative for the evolution of PmatE. The coveted requirements to this new platform are:

- It should include a graphical environment easy to use which allows the teacher to insert parameterized plotting and text.
- The use of a symbolic calculation system in areas such as mathematics, physics or chemistry (equation editor).
- Interface that allows creating and using parameterized questions immediately in the PmatE platform.
- In order to ensure the quality and randomness of the PmatE contents, the new proposal would keep intact the concepts of QGM, *variables* and *boxes of alternatives* or simply *boxes* as traditionally particularly relevant items of QMG.

The concept of box is taken in *ModelMaker* with a friendly visual environment. It allows an additional randomness either in the question or in the answer. Boxes can be explained as dynamic rows of tables, where only one row per table is chosen to concretize. Also, each row may contain inside several boxes (tables) then the final lecture of the random selection can be made from the outside to the inside (see Fig. 1).

c1_1	A maioria da superfície terrestre é ocupada por água.	
c1_2	Na Terra	
	c2_1	existem
		c3_1
c2_2	apenas \$numero\$ oceanos.	
c1_3	O oceano \$oceano1\$ \$tipooceano\$.	
c1_4	O \$tipooceano2\$ oceano da Terra é o oceano \$oceano2\$.	

**Figure 1.** Example of boxes creation

For example, the Fig. 1 shows four boxes (c1\_1), (c1\_2), (c1\_3) and (c1\_4). The box (row) (c1\_2) contains text and also other two boxes (c2\_1) and (c2\_2). Inside box (c2\_1) there are other two boxes: (c3\_1) and (c3\_2). Then, if the box (c1\_2) is chosen, the following possibilities may be concretized:

- *Na Terra existem cinco oceanos*
- *Na Terra existem apenas \$numero\$ oceanos.* (Where \$numero\$ is a variable, that is concretized randomly)
- *Na Terra existe apenas um oceano.*

The remaining of this paper is structured as follows: **Solution Overview** which describes the aims functionalities and steps of *ModelMaker*. **Deployment, evaluation and discussion** which contains a brief description of some results and advantages of the system proposed in this paper. **Conclusions**

and **Future Work** summarizes the results of *ModelMaker* development and elaborates ideas for future work.

## 2 SOLUTION OVERVIEW

The present section describes the main functionalities of the proposed system and provides detailed information regarding its usage.

### 2.1 Model cataloguing

This is the first step to *ModelMaker's* wizard. Fields marked as required can be ignored until the time of posting to the "cloud". Models can only be published when cataloguing is completed.

First, the subject and objectives of the Model must be set. The difficulty level is optional but the "Model Type" must be strictly fulfilled because this specification will contribute to automatically construct structures containing images and text (see Fig. 2).

The screenshot shows a web form titled "Catalogando novo Modelo:". It contains several dropdown menus and text input fields. The "Área Científica" is set to "Matemática", "Área" to "Álgebra e Funções", and "Tema" to "Lógica". The "Sub-Tema" and "Objetivo Principal" dropdowns are both set to "Seleccione SubTópico". The "Objetivo Secundário" field lists "Entidades lógicas", "Operações lógicas", "Relações entre as várias operações lógicas", and "Quantificadores". The "Ciclo de Ensino" field lists "Demonstração", "Método de indução matemática", and "Lógica". There are also dropdowns for "Nível de Dificuldade" and "Tipo de Modelo".

Figure 2. Classification of the model contents, complexity and type.

### 2.2 Text editors

A Model contains an initial text followed by several true/false questions related to the text. To enter the text and the questions of the model we provide dynamical text editors with the essential functions needed for the desired design. These editors are similar to those in Microsoft Word. All selected formatting here such as text color, symbols, fonts and sizes will have the same shape when published (see Fig. 3).

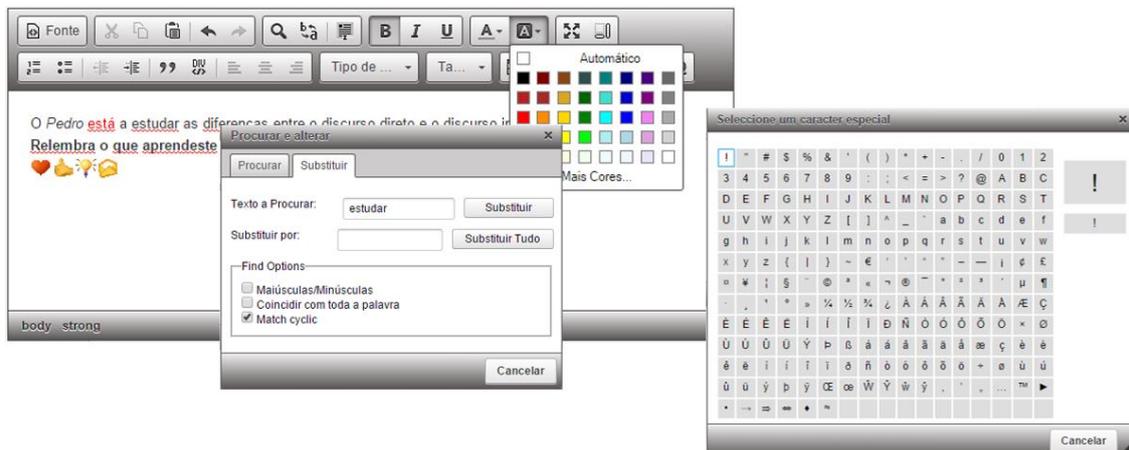


Figure 3. Some functionality of the Text Editors used on *ModelMaker*

### 2.3 Equation editors

It is also provided an "Equation Editor". It includes most of the mathematical symbols, the periodic table of chemical elements, etc. Handling with this equation editor is similar to Microsoft Word equation editor. With a double click, the equation comes into design mode and is updated in runtime (see Fig. 4).

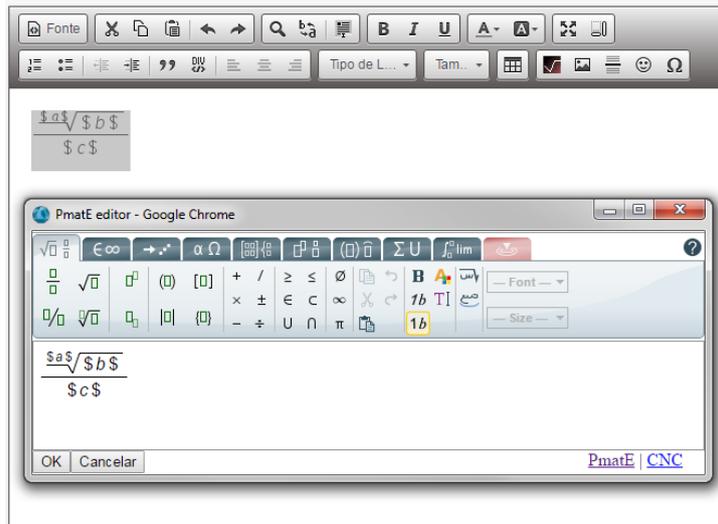


Figure 4. Equation editor used on ModelMaker

## 2.4 Content boxes

Boxes are used to define multiple alternatives, as previously described. By selecting the item "table" we have the following description on the class field: **"boxoptions"**. If we want an item to be contained in a paragraph, then we must complement the class field with **"p"** (like as we do with HTML). If we want to include a line break, then include **"br"** as a complement.

## 2.5 Variables

ModelMaker supports 6 basic variable types: integer, decimal, text, mathematical function, function with dependences and graphical functions programmed with Python.

Each type of variable has a distinct purpose in the Model. At the time of cataloging, ModelMaker application will activate specific steps so that the teacher can fill all necessary information for using these variables. For example, when selecting a variable of type text, a content box will be activated, allowing to insert all possible options using the character "\" as a separator (see Fig. 5).

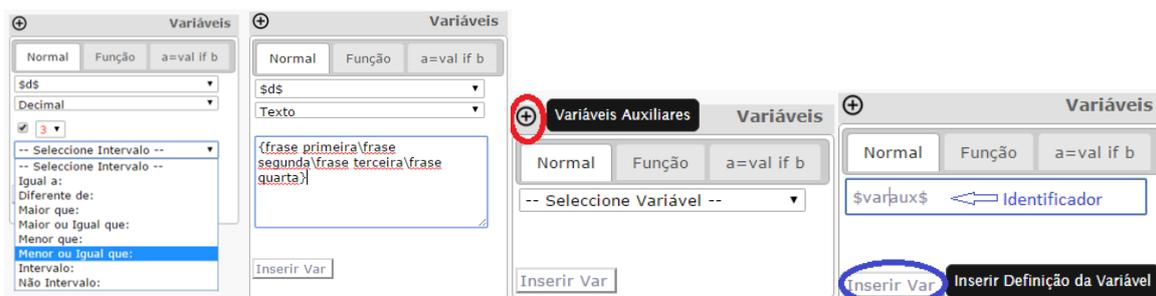


Figure 5. Variables types including auxiliary variables

## 2.6 Mathematic functions

A function is a very useful and dynamic type of variable, available in the system. Functions can be used, recursively, by variables that were previously listed as functions, for example.

The formulas can be inserted on these variables on the second tab in the catalog block. This is done in three steps: a) Entering the formula using the group functions available in Tables 1 and 2 (these function belong to the library NCalc [8]); b) Fix the number of decimal places in the result if necessary, by checking the "checkbox" which gives access to the selector of decimal places (see Fig 6 a); c) If necessary select the "checkbox" to prevent embodiments where the result of the function return repeating decimal values (eg  $1/3 = 0.3333333333\dots$ ; see Fig 6 b)).

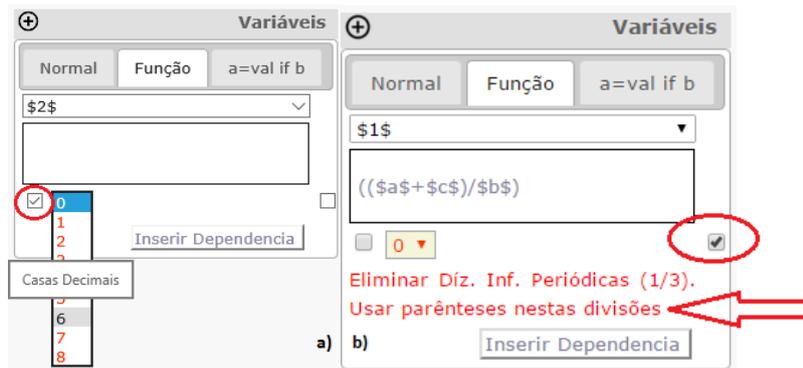


Figure 6. a) Setting decimals places b) Avoiding calculations with periodic results

Table 1. Mathematical functions allowed in ModelMaker

<b>Function</b>	<b>Description</b>	<b>Example</b>
<b>Abs</b>	Returns the absolute value of a specified number.	Abs(-1)
<b>Acos</b>	Returns the angle whose cosine is the specified number.	Acos(1)
<b>Asin</b>	Returns the angle whose sine is the specified number.	Asin(0)
<b>Atan</b>	Returns the angle whose tangent is the specified number.	Atan(0)
<b>Ceiling</b>	Returns the smallest integer greater than or equal to the specified number.	Ceiling(1.5)
<b>Cos</b>	Returns the cosine of the specified angle.	Cos(0)
<b>Exp</b>	Returns e raised to the specified power.	Exp(0)
<b>Floor</b>	Returns the largest integer less than or equal to the specified number.	Floor(1.5)
<b>IEEERem ainder</b>	Returns the remainder resulting from the division of a specified number by another specified number.	IEEERemai nder(3, 2)
<b>Log</b>	Returns the logarithm of a specified number.	Log(1, 10)
<b>Log10</b>	Returns the base 10 logarithm of a specified number.	Log10(1)
<b>Max</b>	Returns the larger of two specified numbers.	Max(1, 2)
<b>Min</b>	Returns the smaller of two numbers.	Min(1, 2)
<b>Pow</b>	Returns a specified number raised to the specified power.	Pow(3, 2)
<b>Round</b>	Rounds a value to the nearest integer or specified number of decimal places. The mid	Round(3.22 2, 2)

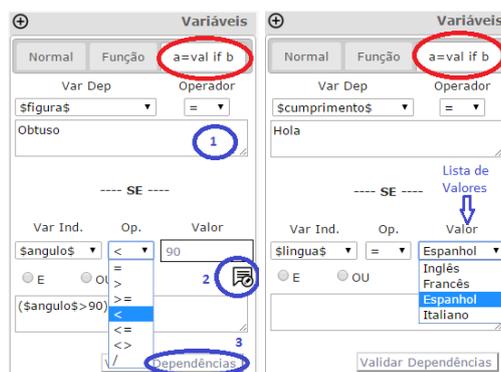
	number behaviour can be changed by using EvaluateOption.RoundAwayFromZero during construction of the Expression object.	
<b>Sign</b>	Returns a value indicating the sign of a number.	Sign(-10)
<b>Sin</b>	Returns the sine of the specified angle.	Sin(0)
<b>Sqrt</b>	Returns the square root of a specified number.	Sqrt(4)
<b>Tan</b>	Returns the tangent of the specified angle.	Tan(0)
<b>Truncate</b>	Calculates the integral part of a number.	Truncate(1.7)

**Table 2.** Login functions inserted in ModelMaker

<b>Function</b>	<b>Description</b>	<b>Example</b>	<b>Result</b>
<b>in</b>	Returns whether an element is in a set of values.	in(1 + 1, 1, 2, 3)	true
<b>if</b>	Returns a value based on a condition.	if(3 % 2 = 1, 'value is true', 'value is false')	'value is true'

## 2.7 Dependencies between variables

The problem of dependencies between variables is one of the most complex challenges to accomplish. It includes many rules and conditions. It is done in two steps: a) insertion of the value / range / "dependent" variable; b) selection of the value / independent interval, when this independent value is of range type. We have implemented mechanisms to minimize typing errors. Figure 7 shows two examples of cataloging dependent variables.



**Figure 7.** Adding dependent variables in science and art cases respectively

## 2.8 SVG Images

The use of SVG images is one of the essential requirements for the presentation of the images in the embodiments and, taking into account that PmatE has a large database of such objects, it was created a search engine to facilitate their reuse on new Models.

For the inclusion of SVG images in Models the process is simple (see Fig. 8). Use the button available in the text editor for image insertion and type the ID field of the desired image that was presented on the search feature.

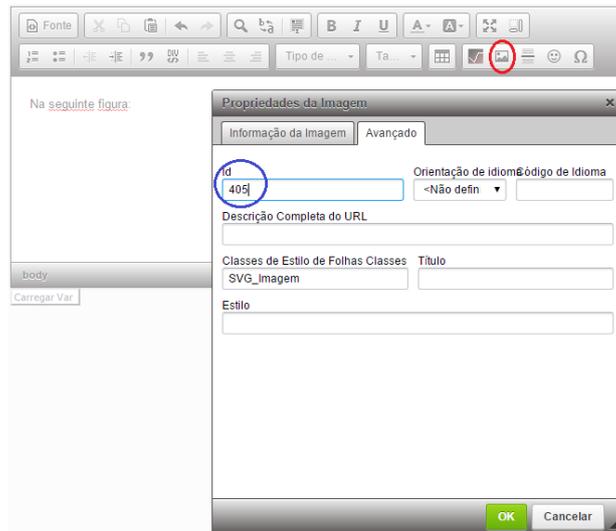


Figure 8. SVG Insertion from Text Editor.

## 2.9 Plotting of functions using python code

The plotting was seen as one of the greatest challenges of PmatE's team. This is the graphical representation of the parameterized variables, which means that in each embodiment a new chart is generated in real time and all images are shown in SVG format.

The plotting is the only functionality you need minimum knowledge of a programming language: Python. It is the best solution we found to plot in real-time with parameters and still get images in SVG format. In general, we can generate any kind of model with images in less than 100ms. Moreover, there are plenty of code examples on the Internet that can be reused and adapted to produce the desired result (seeFig. 9).

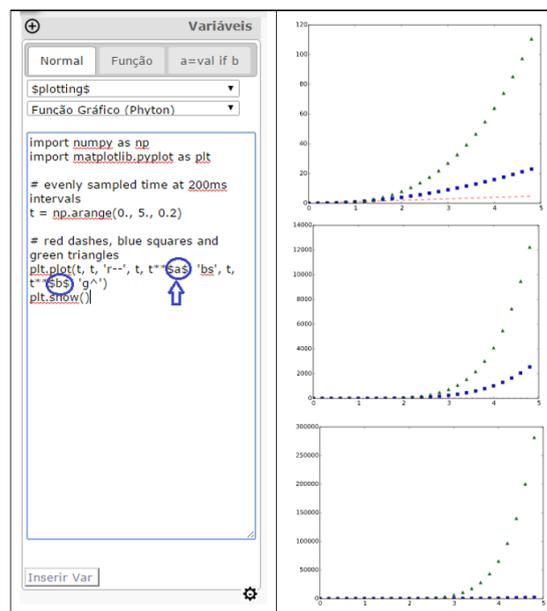


Figure 9. Dynamic Plotting using cataloged variables.

## 2.10 Validation of answers

Together with the initial text and questions, a Model has to provide the validation of the answers, that is, conditions on the parameters defining when each answer is true. For this effect, the logical signs used are: AND; OR; ~; (;); <; >; =; >; <= and intervals. In the case of intervals, we use the command Bet (from between). For example, “(~Bet(\$23\$,4,6))”, means variable \$23\$ is between 4 and 6, inclusive. Text type values must be enclosed in quotes.

There is a shortcut that contains the list of variables, boxes and logical combinations. The teacher can choose to use the shortcut (recommended way) or write directly on the validation text box (see Fig. 10).

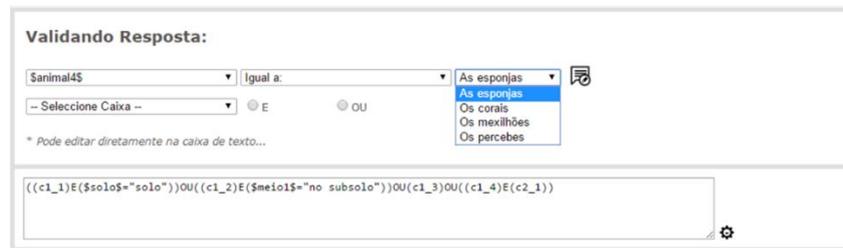


Figure 10. Validator for each response.

### 3 DEPLOYMENT, EVALUATION AND DISCUSSION

In order to assess the performance boost provided by *ModelMaker* as compared to its predecessor, we identified some of the advantages of the newest solution as follows:

- Authors (i.e., the users who create and materialize new Models) no longer require programming or LaTeX knowledge.
- Authors are now able to control directly style parameter which was previously only accessible to the developers, thus removing an intermediate point and ensuring a better and faster match between the desired and the obtained models.
- Programmers are not needed in the production of new Models and can use their time to improve *ModelMaker*.
- The model making process is available through a Web Interface enabling its access to users independently from its knowledge area, geographical locations, etc.
- The system for model evaluation was improved to reduce the assessment time.
- Functionalities such as visualization and save of incomplete models are available.
- Professors have a profile where they can manage their models (e.g., to check which models are currently available on the cloud).
- *ModelMaker* provides access to a database of images which are automatically tagged and can be searched based on those tags.

A comparison between the *ModelMaker* production versus its predecessor is shown in table 13.

Table 3. Comparison between de Models production before and after *ModelMaker*

Comparison Points	Before ModelMaker	After ModelMaker
Level of content complexity	High	Reasonable. Complex mathematics computing is missing.
Time to obtain validated content	Delay of hours or days, depending on programmer availability.	Real time, total autonomy of author
Time to correct content error	Delay of hours or days, depending on programmer availability.	Real time, total autonomy of author
Final results to students	Similar to both versions	

## 4 CONCLUSIONS

- *ModelMaker*, reduced the length and complexity of the model making related tasks, represented a boost regarding productivity at PmatE, as well as autonomy in the Model production in different areas of knowledge. Also, some new functionality was added such as dynamic content trees, unfixed amount of answers per model and a new method to validate contents.
- The development of a suitable entity-relationship model in the database, combined with the use of XML files (for version control per model), gave to this platform robustness in its functionalities.
- It is important to highlight that although Model Maker was conceived for the knowledge areas defined by PmatE, it is generally applicable to other areas which may be evaluated by a QA principle (e.g. Programming, Health or Transit in the learning and evaluation process).

## 5 FUTURE WORK

As future work the *ModelMaker* will be improved in the following three important tasks:

- Addition of advanced mathematical functions. In the case of universities contents, some models need to compute advanced functions that at the moment are not included in this version.
- Addition of parametrized tooltip in the problem resolutions as the best way to the student interaction with the PmatE platform.
- Cloned and Multilingual content using *ModelMaker*. It is an ambition to the team of PmatE project to make International Science Competitions. Then *ModelMaker* would be able to clone concretized contents to different users but in different languages and adapting the Models to different cultures. This idea could suggest a new type of PmatE competition.

## ACKNOWLEDGEMENTS

This work was supported by CIDMA ("Center for Research & Development in Mathematics and Applications") and FCT ("FCT- Fundação para a Ciência e a Tecnologia") through project UID/MAT/04106 /2013 and grant SFRH/BSAB/114249/2016.

The authors are thankful to the PmatE staff members responsible for managing the contents. Their contributions help to improve the ModelMaker functionalities, to identify its limits and contribute with new ideas to future work.

## REFERENCES

- [1] Easygenerator, <<https://www.easygenerator.com/>>, 2016.
- [2] SmartBuilder, Strategic Technology Solutions dba SmartBuilder. <<http://www.smartbuilder.com/product/smartbuilder>>, 2016.
- [3] MLOAT, Multimedia Learning Object Authoring Tool. <<http://www.learningtools.arts.ubc.ca/mloat.htm>>, 2016.
- [4] M.P. Oliveira, S.V. Silva. An overview of PmatE: developing software for all degrees of teaching. Proceedings of the International Conference in Mathematics Sciences and Sciences Education, June 11-14, University of Aveiro, 2006.
- [5] J.C.D. Vieira, Modelo Gerador de Questões, in: P.O. Maria Paula Carvalho (Ed.) Conferência IADIS Ibero-Americana WWW/INTERNET 2004, Madrid, Spain, 2004.
- [6] Anjo, António Batel, M. Oliveira, J. Pinto (2006). Tdmat 9,12 - A diagnostic test in Mathematics for 9th and 12th grades. ICMSE 2006 - International Conference in Mathematics, Sciences and Science Education. University of Aveiro.

[7] Anjo, António Batel; Rui G. Isidro; Paula Oliveira; Sónia Pais; Joaquim Sousa Pinto (2007) TDmat - Mathematics Diagnosis Evaluation Test for Engineering Sciences Students. International Journal of Mathematical Education in Science and Technology 38(3), ISSN: 0020-739X, 283-299.

[8] NCalc, NCalc - Mathematical Expressions Evaluator for .NET. <<https://ncalc.codeplex.com/>>, 2015.