# Agile development of multiplatform educational video games using a Domain-Specific Language

# Agile development of multiplatform educational video games using a Domain-Specific Language

Cristian González García[a]*, Edward Rolando Núñez-Valdez[a], Pablo Moreno-Ger[b], Rubén González Crespo[b], B. Cristina Pelayo G-Bustelo[a], Juan Manuel Cueva Lovelle[a]

[a] *Department of Computer Science, University of Oviedo, Oviedo, Spain.*

[a] *gonzalezcristian@uniovi.es (0000-0002-8810-6023), nunezedward@uniovi.es (0000-0003-4928-4035), crispelayo@uniovi.es (0000-0002-8246-8840), cueva@uniovi.es (0000-0002-7812-7242)*

[b] *Universidad Internacional de La Rioja (UNIR), Logroño, Spain.*

[b] *pablo.moreno@unir.net (0000-0003-4817-8150), ruben.gonzalez@unir.net (0000-0001-5541-6319)*

* Corresponding author, +34985103391

**Keywords:** *Model-Driven Engineering; Domain-Specific Language; Education Tool; Human-Computer Interface; Interactive Learning Environments.*

## ABSTRACT

Educational video games are becoming an increasingly popular alternative in the academic field. However, video game development is a very complex task that requires programming skills and knowledge of multiple technologies, as well as lengthy and costly processes. This has hindered the adoption of educational video games in real settings, and therefore the global acceptance of educational games as a viable approach. In contrast, teachers with limited time and resources are a key factor for educational video game adoption. If enthusiastic teachers can be empowered to create games to engage their students, and if these games were ready to be played in a variety of devices and platforms, we could bring a new generation of low-cost games for immediate deployment in the classroom, using the students' own devices. We aim to achieve this goal by creating a Domain-Specific Language for the development of multiplatform educational video games without requiring any programming skills and with a reduced time investment. This approach should reduce the barriers for using educational video games in the classroom and ease the way towards generalised adoption of educational video games.

## 1 Introduction

The current use of new technologies is widely spread due to the use of Smartphones, the Internet, tablets, or computers [1], for personal use and learning, offering and opening a world of possibilities. This has resulted in schools, high schools, and colleges having a new variety of devices to help students learn. Educators are also becoming more interested in these educational video games to motivate the students and improve learning [2]–[4]. Games help reinforce student motivation when they try to win and they receive emotional benefits when they obtain their reward [5]. According to recent research [6], educators are discovering the cognitive potential of games in the classroom. Furthermore, the use of educational video games is increasing [4], [6]–[8] in fields like health care, soft skills training, math, engineering, business, language, and other domains. We can boost the students' motivation for learning through entertainment software, as proposed in [2], [9]–[11]. Then, through the use of video games, cooperation, competence and self-improvement will be stimulated. This will have repercussions in the individual learning, by trying to surpass other students, and in the team cooperation, by trying to find a combined solution to the problem. The student will also have the possibility of repeating the lessons and exams suggested by the teacher repeatedly. This will prevent students from getting frustrated when failing, because they will be able to repeat the same lessons until they achieve their goals, which will positively improve his motivation for learning [10], [12], [13].

These arguments may announce a new era of education, with a strong presence of technologies [14] and games in the classroom. However, the reality is that games are not that close to global acceptance.

Effective day-to-day classroom innovation works better when adopted by grassroots teachers trying to engage their students. However, teachers cannot be expected to create educational games for their students on their own. There are some educational tools that aim to facilitate this and have been successfully used in the past, including eAdventure (previously known as <e-Game> and after as <e-Adventure>) [11], [15], Scratch [16], [17], and Alice [18], [19]. However, even these tools still demand a lot of time for the development of educational games and can be too much for a single teacher to face alone.

Furthermore, once the games are created, deploying them so that they can be played by a large group of students is challenging. Different schools use different platforms and approaches, and the alternative of using student devices (i.e. BYOD approaches [20]) is even more challenging from a platform perspective, due to the youth and fast evolution of mobile devices and their operating systems. This wide diversity presents additional challenges in terms of developing educational applications that may be supported by all platforms simple and effective way [21].

These difficulties could be tackled by using specific development approaches that target simultaneously different platforms, including mobile platforms. One possible solution would be to create a Domain-Specific Language (DSL) to obtain that abstraction layer and encapsulate the domain of the problem in it [22]. To solve this problem partially, derived from the software development of the application, we will leverage Model-Driven Engineering (MDE).

In this research, we present a DSL that can be used to create educational multiplatform video games. With this DSL, teachers can be enabled to small and simple educational video games in a matter of hours rather than days or months. Teachers can define the different menus, questions, answers, pictures, insert videos and sound and their own images. Teachers can also create different question types: logic, math, trivia, memory, tutorials and other.

While the games themselves are relatively simple, the process is so straightforward that a teacher may create a small game focused on current events in one evening so that students can play it the following day in any platform. The DSL also provides an abstraction of the actual game platforms and look & feel settings, therefore creating games that can be deployed to many different platforms without a significant effort.

The remainder of this paper is structured as follows: In section 2, we analyse the current situation of the development of applications for mobile devices and make a brief explanation of Model-Driven Engineering and Domain-Specific Languages, as well as research on some existing video game editors and the situation of video games in education. Section 3 details the kind of questions that can be created in games and the implementation of our DSL in our Case Study. Section 4 explains all the methodology and the results of the experiment. Section 5 contains the discussion of the research and the future work.

## 2 STATE OF THE ART

Educational video games could be a good tool to motivate students [4], [23], giving positive effects on them. Nevertheless, not all people have programming skills or know how to create a video game and then people would require time for learning about these things. Furthermore, another challenge is the necessity of a high investment because game development requires high development costs [24].

Therefore, a possible solution would consist in facilitating the development of educational video games for teachers with a DSL which facilitates the game development. This is why we chose to abstract the problem using MDE to create the DSL. With this DSL, teachers could create educational video games with no programming skills, but in an easy and quick way as we can see in literature [25], including lessons and the corresponding tests.

### 2.1 *Development of educational applications for mobile devices*

The wide range of electronic devices in the student's daily life offers endless possibilities. These devices allow to do the exercises and learn by playing anywhere, as long as they have smartphones, laptops, tablets, or any other similar devices, even in places with few technological means or where the access to the Internet is not possible [26]. But, because of the great variety of electronic devices, it becomes very difficult to develop educational applications for all kind of devices efficiently [21]. This is due to the big differences that exist between the different mobile operating systems in the market. For example, Android uses Java, and iPhone uses Objective-C or Swift [27]. Furthermore, evolution in the mobile world is fast and constant.

As we have seen, game development is very expensive and needs educational experts to help in the creation of the game. However, teachers do not have much time and usually no external incentives to participate in game design [28]. It is a reason why they need tools to facilitate the development of educational video games in an easy and quick way with a low individual cost [29].

### 2.2 *Video Games in Education*

Learning through video games is nothing new. For instance, video games are able to motivate the students and encourage them to solve problems, reflect, and think in a more creative way. This makes students learn through participation instead of memorisation [12]. There is also significant evidence that video game players improve their reaction times, their hand-eye coordination, and their self-esteem [30].

We can see in literature as some teachers have used AAA video games in education. For example, with Age of Mythology, students learned about various mythologies from the ancient world and Age of Empires contained information about ancient civilisations, heroes, weapons, wars, and history [31]. With Civilization IV it was demonstrated that students learned about the history of civilisations by giving life to all kind of ancient empires [9], or to teach politics, civics, and history [29]. There are also economy games, like Patrician and Pharaoh, in which the player has to manage an empire, check the needs of the citizens, and raise money, something that helps the players to understand better the way an economy worked back then, as well as experiencing a whole history lesson while playing an entertaining game. Other games, like SimCity 2000 are useful to learn about urban geography [10] or social dynamics and evolution [23]. Other games depend on the player to make some moral decisions on the entrusted requests in order to advance through the game, like what happens in Star Wars: Knights of the Old Republic [31]. Other ones allow players to create their own map for the game, with the possibility of generating things and not only learning, which is what usually happens in schools [32].

On the other hand, the research literature on (non-commercial) educational video games shows that they have been successfully applied in multiple fields, including high-stakes areas such as reducing the errors of health professionals [33], [34]. Other examples are: [35] that obtained an increment of the average student note in the midterm exam and reduced the student stress and increased their satisfaction; [36] that compared the game-based learning with the traditional methods with undergraduate students, with an increment of the mean score in the first group; while [37] researched that the students that used the game to learn had obtained a higher level of their cognitive process and more satisfaction than students using traditional methods.

In other cases, the games were used for obtaining benefits in some specific users, improving their memory, reflexes, and knowledge. For instance, educational video games for toddlers, so they can learn basic things, like colours or the alphabet. Games could be used to help elderly people [38] too, improving and reinforcing the technique or strategies with health professionals [33], [34]. Besides, video games can help people with some kind of disabilities [39] or disorders, for example on kids with Attention Deficit Hyperactivity Disorder (ADHD) [30], hearing impairment [40], between others as we can see in this systematic review [33]. Even, games can be aimed to stimulate specific capacities [7], such as: memory, logic, eye-hand coordination, reaction time, self-esteem, spatial vision, etc.

### 2.3 *Model-Driven Engineering*

Model-Driven Engineering (MDE) appeared to solve software development problems [41], to be precise, the problems of the Software Crisis, which was already present in the 1960s and are still present, as can be seen in [42], [43], in spite of the use of agile approaches [44]. One solution to these problems can be obtained through the automation or semiautomation of processes, something in which MDE is quite popular for solving different problems [45]–[49].

With the use of MDE we manage to reduce the complexity of the design and the implementation, which helps to obtain a much more reliable software and with more sophisticated functionalities [50]. Using MDE, we can increase the abstraction over third generation programming languages (e.g., C++, C#, Java, and so on). This offers the use of a concept much closer to the problem domain by converting the elements of the domain into one or more models. This model makes easier for us to create a DSL that we manage to increase the abstraction of the problem, giving us an increase of the productivity [50], [51].

By applying MDE, we manage to simplify the abstraction level for the different generated educational video games. This gives us the necessary abstraction to create the DSL that allows people without knowledge the creation of simple educational video games in an easier way for teachers, also being able to port these games to target different platforms, one of the main objectives of this research.

### 2.4 *Domain-Specific Language*

A Domain-Specific Language (DSL) is a language, commonly declarative, that generates calls for subprocesses in order to solve a certain problem within a specific domain. DSLs have a great power of expression [22]. The main advantages that the use of a DSL can offer are the increase of the productivity, the lesser chances of errors, easy maintenance [52], [53], portability, the knowledge of the problem domain, and the reutilisation for different purposes [22], [54]. On the other hand, DSLs present handicaps like a worse efficiency than native codification and a higher difficulty to create the domain and construct the DSL [22], [52]. However, DSLs can be created applying MDE to solve a problem [51], thus creating an abstraction level of software engineering [55].

By providing a DSL, it is possible for any person to define the desired educational video game, which is our domain, in a quick and simple way, creating a supported language for the different platforms, which guarantees the reusability of all the elements they share.

### 2.5 *Video Game Editors*

Currently, several tools make it easier to develop video games (educational or not). Some tools allow exporting the game to various platforms at the same time, making it possible to develop the game once and then running it in multiple platforms, The biggest differences amongst the different editors are: the type of games supported (graphic adventure, simulators, arcades, etc.), the possibilities for editing and expanding existing video games, the usability of the tools themselves, and the straightforwardness they offer when creating a video game. In this section, we will describe some of the editors used by collegues from other research projects.

**eAdventure Project** is a research project that has been created and it is maintained by e-UCM research group from Complutense University of Madrid [15]. The first prototype was developed by [56] and it was called **<e-Game> Project**. The project offered a DSL to create adventure video games to motivate their students. Around 2006-2007 it was renamed to **<e-Adventure>** [21], [57], [58]. It allowed developing the video game by teachers without the necessity to have a development team to help teachers during the process. Furthermore, the development cost for educational video games was drastically reduced. However, it only supports adventure-like video games.

**GameMaker Studio** allows developing multiplatform video games quickly [59]. It was designed to be used by inexperienced users [60]. This editor allows creating the levels and menus of the video game in a graphic way, by dragging the elements where we want. It also allows programming in its own language, GameMaker Language, which requires some basic programming knowledge. Notwithstanding, the editor generates an executable file, which does not allow further modifications, it does not offer special help to create educational video games, and it has very expensive licenses.

**GameSalad** is a free software tool that allows the user to develop and export games for Mac, PC, iOS and Android devices, and HTML5 [61]. In [13], it was used to create a video game to teach wind and gravity theory on iOS. In [62], it was compared with App Inventor for Android. GameSalad offers a series of advantages that makes video game development easier using a drag-n-drop program creation feature, defining actors and scenes, and configuring their parameters to establish their design, behaviour, and the game rules [62]. However, we consider it a limited tool because the created video games can be modified only from the tool itself and GameSalad does not have special templates to facilitate the development of educational video games.

**Stencyl** is a tool that provides video game designers with a graphic editor that enables game creation for iOS, Mac, Windows, Flash, Android, and HTML5. Stencyl is also based on the definition of scenes and actors. Nonetheless, Stencyl is intuitive enough for people that have experience with design programs. However, programming and algorithm logic knowledge is necessary, as many of the modifications done to a game require the creation of complex flow diagrams.

**Unity** is a video game engine developed by Unity Technologies [63] that allows the creation of 2D and 3D multiplatform video games. It features a graphic interface that allows dragging components and customizing them with the editor. This allows modifying the physics, graphics, behaviours, and interactions of the elements, and allows importing object models created with graphic design tools. It also has its own scripting language to program actions in video games, adding behaviours, and making the objects more customizable [64]. However, once the game is generated, the editor gives a compiled solution, which makes impossible to edit the game or its levels without using the editor. Besides, Unity does not have special templates to facilitate the development of educational video games.

## **3** CREATING THE DSL

In this research, we have implemented the DSL in the **Gade4All** project. **Gade4All** was a national research project financed by the AVANZA plan and performed by the 'Ingeniería Dirigida por Modelos MDE-RG' research group at the University of Oviedo, which resulted in the creation of a 2D native multiplatform video game editor.

In [65], [66], we can see the basic idea about the editor and how Gade4All works to create platform video games and the idea of this project. In these first articles, the main idea of the different typologies was presented along with the design templates for the menus, how they work, their creation, and the editor architecture. In this work, we focus on the DSL itself, the game-building process and the results of an evaluation of the overall agility of the process.

In this new iteration of our research, we implement the necessary logic to develop educational video games based on questions.

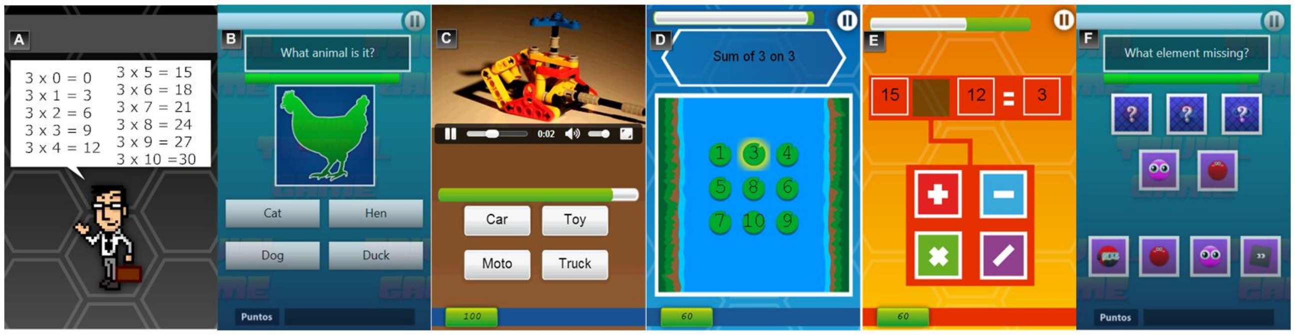


**Figure 1 Montage featuring examples of the different types of questions**

This current iteration contains a DSL, which allows creating simple educational video games with educational questions in an easier way. From a software engineering perspective, we used Model-Driven Engineering to abstract the problem, create the DSL, and make easier the video games portability among platforms. One of the main pillars of this platform is the creation of a DSL that allows defining the main characteristics and working on the games taxonomies in a specific and non-platform-dependent way. The implementation of this DSL in Gade4All allows creating multiplatform educational video games using the Gade4All editor with a minimum effort for teachers.

The games first offer an educational part and then ask questions about its content. However, this can be manipulated at will by placing the information and the questions in the order that we prefer. After getting to the end of the game, the player immediately obtains his score. All the generated educational applications run on mobile devices (Android, iPhone) and in any web browser with HTML5 support.

## 3.1 *Designing the games*

In this subsection, we will describe, in general terms and with a high level of abstraction, the possible kind of questions that can be created. We show an example of questions in Figure 1. The questions can be divided into six groups: tutorials, mathematics, trivia, mental agility, interaction, and software. All the questions share common elements and any kind of question can have multiple answers, include sounds and videos and add properties from another type of questions. There is also a time meter that can be adjusted individually for each question in order to establish a time limit.

However, any type of action and customisation for a question is applicable to any other. For instance, every question can contain video and audio and the actions performed in the interactive questions or the mental agility ones can be performed even in the tutorial question. Still, these six groups are merely indicative, because the customisation possibilities allow the user to create new kinds of questions:

**Tutorials:** As it can be seen in Figure 1A, the teacher can include the information that he wants to give to the student. Like the normal questions, this is totally customizable and multimedia components can be added, like video and audio files. This way, the teacher will be able to teach the student by creating one or more tutorials, followed by an educational video game.

If implemented, the audio and video files can be played, paused and stopped at will by the students. In this case, it is possible to offer an audio tutorial to make things easier for the students and let them choose between reading and listening.

**Trivial:** Figure 1B and Figure 1C show examples of trivia-like questions. One features an image and the other a video. This kind of question can contain both images, texts, videos and sounds, so the teachers are given a lot of freedom to create their own questions. For instance, the teachers can include images of pictures and drawings in their questions about history and art, or images of clocks to ask the students what hour is displayed on them, maps and flags of countries for the geography lessons or even sound files for the questions about music. Regarding the text of the question, the teachers can write their own text, including numbers and mathematical signs. Another possibility would be to include videos for the creation of cultural games about movies, documentaries, history, theatre, interviews or videos of experiments performed in the physic and chemistry lessons, even videos of exercises for physical education lessons.

**Interaction:** Teachers can also create interactive questions. As it can be seen in Figure 1D, it is possible to create a sequence of additions so the student selects the proper way to reach the last number. Another example of an interactive question could be a geographic map in which the student would have to search for a certain country or province. With this kind of functionality, the student is motivated to perform a task that would normally be boring in a more entertaining way.

**Mathematics:** As seen in Figure 1E, the teacher can help the students to practice and improve their calculation capacity and mental agility. In these, the answers can be a number, an operational symbol, or both, if we choose to create a multi-answer question. For the aforementioned, there is the option of creating more complex operations by adding brackets and square brackets. There can be many possible right answers that have to be selected.

**Mental Agility:** Teachers can configure the elements on the screen so they disappear after a small amount of time. Because of this, they can create questions that allow the students to test their mental agility. Figure 1F is a good example of this, as it features a memory question, where three elements are shown, later they are replaced by question marks, and a second set of images appear, showing only two of the three initial elements. Some seconds later, the next four answers appear and the students must choose the element of the first set of images that is missing in the second one.

Other examples of this kind of question could be: give the image of an element and ask the students to find it in a collage with a similar image, count the number of times that an element appears in an image or search the differences between two very similar images.

## 3.2 *Implementation of the Domain-Specific Language*

To make the process of porting the educational application between platforms easier, we used MDE. For this reason, the architecture of the system can be divided into two big groups. The first one is the created DSL to abstract the code that is common to all platforms. The DSL allows defining all the information about the menus and levels of the educational video games and it is the same on all the platforms. In this way, we generate a language that can be interpreted by the templates in native language that we developed for each device. Then, after the teachers define their educational video game using our DSL, they obtain the model of their game. The model has all the information that the user had defined using the DSL, and the DSL allows creating with an abstraction the video games. The model consists of two XMLs. These XMLs files contain the information of the menus and levels of the educational video games. These XMLs are common to each platform, so the same level/menu file can be used in any platform.

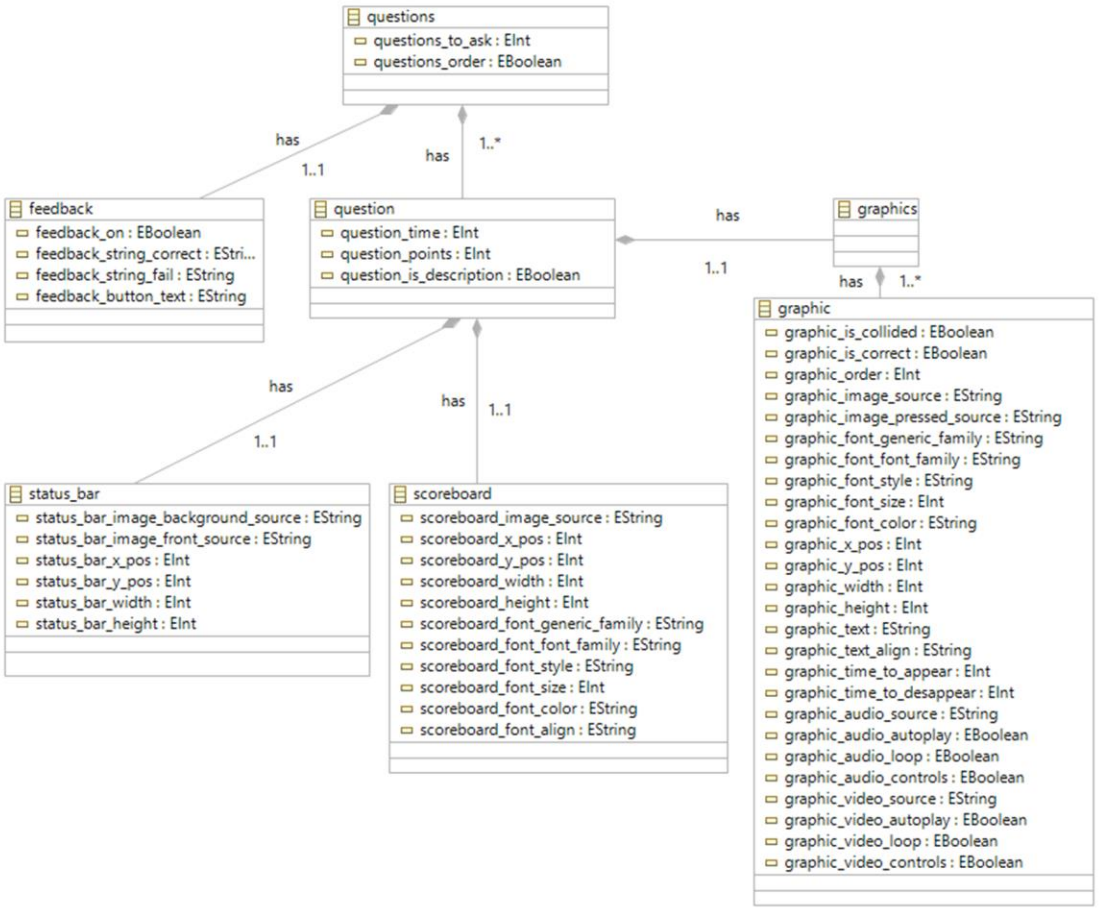
The second group are the templates. They contain the native source code for each platform and the parser of the files that contains the information that a user defined using our DSL. The templates do not have the images, nor the information about menus, nor levels, which means that they cannot be compiled without the DSL nor the resources.

### ***3.2.1*** *Abstract Syntax*

In this section, we show one part of the abstract syntax of our DSL. We use Ecore [67] as meta-metamodel to create the metamodel of the DSL. In Figure 2, we can see this first metamodel of the part that corresponds with the levels. The principal component is *questions*. It wraps all the data that define a game level and it contains the general information about the level: number of questions and the order.

*Questions* has two child nodes: *feedback* and *question*. The first one, *feedback*, shows the information that will be shown when answering a question, if desired. The second one, *question*, wraps all the information about each question.

The *question* node is composed of three general parameters and three child nodes: *status_bar* that defines the aspect and position of that bar, *scoreboard* which defines the image used for the scoreboard, and *graphics* that contains all the graphics that composed the question. Each *graphic* contains the parameters that define it, for example: images, font, audio source, or video source.



questions
questions_to_ask : EInt
questions_order : EBoolean
has
1..1
has
1..*
feedback
feedback_on : EBoolean
feedback_string_correct : EStri...
feedback_string_fail : EString
feedback_button_text : EString
question
question_time : EInt
question_points : EInt
question_is_description : EBoolean
has
1..1
graphics
has
1..*
graphic
graphic_is_collided : EBoolean
graphic_is_correct : EBoolean
graphic_order : EInt
graphic_image_source : EString
graphic_image_pressed_source : EString
graphic_font_generic_family : EString
graphic_font_font_family : EString
graphic_font_style : EString
graphic_font_size : EInt
graphic_font_color : EString
graphic_x_pos : EInt
graphic_y_pos : EInt
graphic_width : EInt
graphic_height : EInt
graphic_text : EString
graphic_text_align : EString
graphic_time_to_appear : EInt
graphic_time_to_desappear : EInt
graphic_audio_source : EString
graphic_audio_autoplay : EBoolean
graphic_audio_loop : EBoolean
graphic_audio_controls : EBoolean
graphic_video_source : EString
graphic_video_autoplay : EBoolean
graphic_video_loop : EBoolean
graphic_video_controls : EBoolean
has
1..1
has
1..1
status_bar
status_bar_image_background_source : EString
status_bar_image_front_source : EString
status_bar_x_pos : EInt
status_bar_y_pos : EInt
status_bar_width : EInt
status_bar_height : EInt
scoreboard
scoreboard_image_source : EString
scoreboard_x_pos : EInt
scoreboard_y_pos : EInt
scoreboard_width : EInt
scoreboard_height : EInt
scoreboard_font_generic_family : EString
scoreboard_font_font_family : EString
scoreboard_font_style : EString
scoreboard_font_size : EInt
scoreboard_font_color : EString
scoreboard_font_align : EString

**Figure 2 Abstract Syntax of the DSL**

```
<?xml version="1.0" encoding="utf-8"?>
<questions>
    <questions_to_ask>22</questions_to_ask>
    <questions_order>YES</questions_order>
    <feedback>
        <feedback_on>YES</feedback_on>
        <feedback_string_correct>Is correct</feedback_string_correct>
        <feedback_string_fail>You failed...</feedback_string_fail>
        <feedback_button_text>Next</feedback_button_text>
    </feedback>
```

**Source Code 1 Example of game that contains the tags that define the feedback for an answer**

### *3.2.2 Textual Concrete Syntax*

In this section, we explain the textual concrete syntax of the XMLs. Firstly, we show the XML of levels. This XML is composed by three parts:

Source Code 1 shows the first part, the parent node: **questions**. Within this node all the questions are stored under the **question** tag and the information about the quantity, order, and feedback parameters shown when answering a question. This node also contains data that is common to all the questions of that level. These are the **question_to_ask**, **questions_order** tags and the child node **feedback**.

The **question_to_ask** tag indicates the total number of questions the level has. In this number are included the tutorial screens and it can show an inferior number to the total amount of questions included, if we do not want to show all the questions included in the aforementioned level.

The next tag is called **questions_order**. This tag indicates if the questions that we will be asked will be shown in the same order that is specified in the XML (*YES*) or if they are shuffled every time we play the level (*NO*). If the **question_to_ask** tag does not contain the total number of existing questions in the XML file, we can create levels with a great amount of questions presented in order. This way, we can create different exams and tests for each student using the generated educational video game.

```
<question>
    <question_time>50</question_time>
    <question_points>75</question_points>
    <question_is_description>NO</question_is_description>
    <status_bar>
<status_bar_image_background_source>timebar.png</status_bar_image_background_source>
<status_bar_image_front_source>greentimebar.png</status_bar_image_front_source>
        <status_bar_x_pos>160</status_bar_x_pos>
        <status_bar_y_pos>240</status_bar_y_pos>
        <status_bar_width>200</status_bar_width>
        <status_bar_height>20</status_bar_height>
    </status_bar>
    <scoreboard>
        <scoreboard_image_source>buttongreen.png</scoreboard_image_source>
        <scoreboard_x_pos>60</scoreboard_x_pos>
        <scoreboard_y_pos>460</scoreboard_y_pos>
        <scoreboard_width>100</scoreboard_width>
        <scoreboard_height>50</scoreboard_height>
        <scoreboard_font_generic_family>serif</scoreboard_font_generic_family>
        <scoreboard_font_font_family>courier new</scoreboard_font_font_family>
        <scoreboard_font_style>italic</scoreboard_font_style>
        <scoreboard_font_size>15</scoreboard_font_size>
        <scoreboard_font_color>black</scoreboard_font_color>
        <scoreboard_font_align>left</scoreboard_font_align>
    </scoreboard>
```

**Source Code 2 Example of game containing the tags that define a time meter and a scoreboard**

The children node **feedback** is composed of four tags that configure the feedback: **feedback_on** can be activated or deactivated after answering a question. With **feedback_string_correct** we specify the chain of text that we want to display when a question is answered correctly. With **feedback_string_fail** we specify the chain of text that we want to display when we choose a wrong answer. For the text in the button, we use the **feedback_button_text** tag.

Source Code 2 shows an example of the first part of the **question** node and the graphic example corresponding to the code shown in the status bar and the scoreboard. This one is composed of three subsections: The children of the **question** node, the **status_bar** children, and the **scoreboard** children. **Question** has three tags that are used to define the time limit for the question (**question_time**), the points received for a correct answer (**question_points**) and if it is a tutorial or a normal question (**question_is_description**).

```
<graphics>
    <graphic>
        <graphic_is_collided>NO</graphic_is_collided>
        <graphic_is_correct> </graphic_is_correct>
        <graphic_order> </graphic_order>
        <graphic_image_source>question.png</graphic_image_source>
        <graphic_image_pressed_source> </graphic_image_pressed_source>
        <graphic_font_generic_family>sans-serif</graphic_font_generic_family>
        <graphic_font_font_family>times new roman</graphic_font_font_family>
        <graphic_font_style>italic</graphic_font_style>
        <graphic_font_size>20</graphic_font_size>
        <graphic_font_color>white</graphic_font_color>
        <graphic_x_pos>160</graphic_x_pos>
        <graphic_y_pos>78</graphic_y_pos>
        <graphic_width>278</graphic_width>
        <graphic_height>86</graphic_height>
        <graphic_text>What is the capital of Spain?</graphic_text>
        <graphic_text_align>center</graphic_text_align>
        <graphic_time_to_appear>0</graphic_time_to_appear>
        <graphic_time_to_desappear>0</graphic_time_to_desappear>
        <graphic_audio_source> </graphic_audio_source>
        <graphic_audio_autoplay> </graphic_audio_autoplay>
        <graphic_audio_loop> </graphic_audio_loop>
        <graphic_audio_controls> </graphic_audio_controls>
        <graphic_video_source> </graphic_video_source>
        <graphic_video_autoplay> </graphic_video_autoplay>
        <graphic_video_loop> </graphic_video_loop>
        <graphic_video_controls> </graphic_video_controls>
    </graphic>
```

**Source Code 3 Example of game containing the tags that define a graphic belonging to a question**

If we want to create a tutorial screen (for example, an explanation of the multiplication tables) the **question_is_description** tag must contain *YES*. In that case, points and time will not be taken into consideration.

The second part is the **status_bar** children, which is composed of two images, the back image (**status_bar_image_background_source**) and the front image, which is the one that becomes smaller as time passes by (**status_bar_image_front_source**). The other tags define the central position of the image in the x-axis (**status_bar_x_pos**) and the y-axis (**status_bar_y_pos**), as well as the width (**status_bar_width**) and weight (**status_bar_height**) of both images.

In the third part is the **scoreboard** node. This defines the image used for the scoreboard (**scoreboard_image_source**), its central position in the x-axis (**scoreboard_x_pos**) and the y-axis (**scoreboard_y_pos**), as well as its width (**scoreboard_width**) and height (**scoreboard_height**). This node also has tags that define the font, family, style, colour, size of the text, as well as its horizontal alignment regarding the image that contains it.

Source Code 3 shows the code of the second part of the **question** node. The rest of the **question** node is formed from the **graphics** node. Within it, we must place the **graphic** nodes in the order they are painted. The first to appear will be the first to be painted, so if the second is painted over the first, it will be screened. Each graphic node is a new layer.

This type of node contains all the required information to paint a graphic on the screen. Then, a graphics group of nodes conform a question. The great amount of tags it has and the variety of them, makes this node the most complex of all. This is because in the game everything is a graphic: backgrounds, buttons, images, sounds, videos, etc. This way we have the required power to put all the workload in the auto-generated part that is contained in the templates (the logic), making the creation of games easier and more intuitive.

The first thing to do is point out if the graphic is collisionable (**graphic_is_collided**). This way we can decide which elements we can interact with, as well as creating elements that change their image when they are clicked. To distinguish a correct answer from a wrong one we use the **graphic_is_correct** tag. With the **graphic_order** tag we can define the order in which the graphics must be clicked. For instance, if we display a set of numbers and we want the student to click them in a certain order, we must assign a number to each graphic to define that order. If we do not want to establish any order, we simply leave this field empty.

To assign an image to a graphic, we have to put its name in the **graphic_image_source** tag. If we want this image to change after a while or when the graphic is selected, we have to place that new image in the **graphic_image_pressed_source** tag. As with other tags, the central position of the image regarding the x and y axes, its width and height are defined in the corresponding tags: **graphic_x_pos, graphic_y_pos, graphic_width** y **graphic_height**.

To define the text of the graphic it must be written in the **graphic_text** tag. The text will be aligned to the vertical centre of the image by default. This tag supports carriage returns, so it is possible to leave blank spaces to fill them with images or more text. As the previous nodes, here we can configure the text size, font, style, colour, alignment, etc. With the **graphic_time_to_appear** and **graphic_time_to_desappear** tags, we can make the graphic appear or disappear after some time or even both. This way, we can create questions where a graphic appears after X seconds while others disappear. In the given example some graphics appear by default in the second 0 and after a few seconds they disappear while others appear.

Lastly, we have the tags that define the sound and the video, which require four tags to control all the options. First, we change the name of the file with the **source** tag. The file can play automatically as the question starts (**autoplay**), be restarted when it ends (**loop**) or show its audio/video controls (**controls**).

```
<number_of_levels>7</number_of_levels>
<screen_width>320</screen_width>
<screen_height>480</screen_height>
<screen_orientation>portrait</screen_orientation>
<menu_view>
    <image_start_button>button.png</image_start_button>
    <start_button_align>center</start_button_align>
    <start_button_size>20</start_button_size>
    <start_button_generic_font>courier new</start_button_generic_font>
    <start_button_color>white</start_button_color>
    <height_start_button>78</height_start_button>
    <width_start_button>230</width_start_button>
    <pos_x__button_start_menu>160</pos_x__button_start_menu>
    <pos_y_button_start_menu>270</pos_y_button_start_menu>
    <start_button_text>Play</start_button_text>
    ...
</menu_view>
<music_view>
    ...
</music_view>
<pause_view>
    ...
</pause_view>
<select_level_view>
    ...
</select_level_view>
<play_again_view>
    ...
</play_again_view>
<end_game_view>
    ...
</end_game_view>
<images_levels>
    ...
</images_levels>
<loading_view>
    ...
</loading_view>
```

**Source Code 4 XML containing the tags that define the menus and levels used in the game**

In Source Code 4, we can see the menus' XML. This XML contains all the information about the different screens and the game's global characteristics: the information about the total number of levels of the educational video game (**number_of_levels**) and the width (**screen_width**) and height (**screen_height**) of the screen in which it was generated. This is because, aside from being multiplatform, the generated educational video games fit the screen of any device. That is why it is necessary to indicate the width and height of the screen for the correct functioning of the resolution adapter. With the **screen_orientation** tag the game changes the visualisation of the screen from vertical to horizontal and vice versa.

The rest of the XML is composed of two children nodes. Each node is a screen of the game. The main menu (**menu_view**) is where the student chooses between playing the game, changing the options, or exiting the game. The pause screen (**pause_view**), as its name suggests, is for pausing the game in the middle of a question. The level selection screen (**select_level_view**) allows the student to select which level he wants to play. The 'play again' screen (**play_again_view**) appears after checking the score for a level and lets you choose between replaying the level or going back to the level selection screen. The screen that appears after a level is completed (**end_game_view**) shows the score for that level. The loading screen (**loading_view**) appears while the game is loading. There is also an **images_levels** node, which contains the images that represent each level in the level selection screen. Each node contains all the customizable elements of the screen it belongs to. For instance, **menu_view** contains the image of the *Play* button, as well as the text of that button, with all its characteristics. The same happens with the rest of elements: *Options* button, *Exit* button and the background image. The remaining images follow the same standard: each element on the screen read from the XML because every element is customizable because of the templates of each platform.

### *3.2.3 Templates*

The templates were created through an educational video game natively created in each platform: Android, iPhone, and HTML5. The only difference with that native video game is the absence of the models that contain the information that the user defined using our DSL, images and videos. At the beginning of the project, we studied the four systems to see the possibilities that each one offered. After this, we started creating the first template for the most restrictive system. Then, we ported it to the other systems, adequating everything to the different programming languages and the possibilities of the systems to paint images, display text, and play audio and video. In this way, to convert the templates for the desired system, we only have to insert the models that we created using our DSLs, images, sounds and videos in their folders, and then compile everything.

### *3.2.4 Software*

To create the templates for the educational video games in each platform, we used the proper environment for each case:

- To create the game template for Android mobiles, we used the Software Development Kit (SDK) for Android.
- For iPhone, we used xCode and a Mac OS X operating system with Mavericks.
- For the HTML5 template, we simply used a text editor and the latest version of the Apache web server. In addition, we checked its proper operation and performed the tests with the most used Internet browsers: Google Chrome, Internet Explorer, Firefox, Opera, and Safari.

The smartphones used to test the Android, and iPhone versions were different in size and worked with different operating system.

## 4 EVALUATION AND DISCUSSION

In this section, we describe in detail the processes of evaluation selected and then we describe the obtained results. First of all, in the Methodology subsection, we will describe the used methodology to perform the evaluations. Finally, we will describe and discuss the obtained results.

### 4.1 *Methodology*

To test the performed research, we combined a formative and a summative evaluation. On the one hand, we wanted to see how users would use the language in a real unassisted setting in order to find bugs and syntax shortcomings. On the other hand, we wanted to get a clear idea of how close we were to our goal of allowing agile game development in the scope of hours rather than days or months.

With these goals, we started by creating four fully functional games in native code for each platform. Secondly, we tested the ecosystem with students, developers, researchers, engineers, and PhDs at University of Oviedo.

The primary objective of the first phase was the validation of our hypothesis: User can create an educational video game faster with our DSL than if he had developed the video game with their own code. Furthermore, we could obtain a lot of information to improve the DSL and the templates. With this information, we changed some syntax of the DSL. For example, we introduced new attributes, changed some nodes, and included new options for some nodes. Moreover, we corrected some errors and bugs.

The second phase also took place at the University of Oviedo. This test was a contest open to every student and teacher, in which the contestants had to create a video game using the Gade4All editor. In the contest, the participants created different educational video games with the Gade4All editor. In addition, we could test a new version that included the Improvements that we realised after the first phase, and the DSL with new and more users to find new bugs and suggestions.

In both phases, we took note of the errors that popped up and the things that were not very clear for the participants when they were creating their educational video games. Each user had a personal assistant for the test.

Firstly, the assistants explained the goal of the project and the objective of the test. Besides, the assistants were taking notes about all the process to make easier the identification and localisation of any error, type of problem (user interface, DSL, model, code generation, template, Gade4All tool) or suggestion.

When the test finished, we compiled and studied all the notes to improve the DSL for the final version.

### 4.2 *Results*

The creation in native code of the four sample games required between 80 and 125 hours (see Table 1) depending on the platform. Then, we recruited 15 participants (1 to 15) to recreate those games using the DSL, and each game was completed in less than 1 hour.

In the second phase, we had 15 (16 to 30) new and different participants: 10 students and 5 teachers. The participants created educational video games with the version of the new tool without the bugs, which we had detected in the previous phase.

In this test, users only needed again between 30-60 minutes to create their own educational video games. The main difference with the previous phase was the improvement of the DSL and the tool.

Table 2 shows the time that each participant needed to finish the educational video game and the total time and average of each phase. In this last table, we can see that in the second phase the participants needed less time because we had solved some bugs.

|  | Android | iOS | HTML5 | Total |
|---|---|---|---|---|
| Hours | 125 | 113 | 86 | 324 |

**Table 1 Required time to create each video game**

| Phase 1 |  | Phase 2 |  |
|---|---|---|---|
| P1 | 46 | P16 | 46 |
| P2 | 59 | P17 | 31 |
| P3 | 57 | P18 | 32 |
| P4 | 52 | P19 | 50 |
| P5 | 51 | P20 | 54 |
| P6 | 39 | P21 | 43 |
| P7 | 49 | P22 | 51 |
| P8 | 44 | P23 | 42 |
| P9 | 56 | P24 | 52 |
| P10 | 43 | P25 | 47 |
| P11 | 48 | P26 | 39 |
| P12 | 41 | P27 | 46 |
| P13 | 58 | P28 | 43 |
| P14 | 56 | P29 | 35 |
| P15 | 44 | P30 | 49 |
| Total Phase 1 | 743 | Total Phase 2 | 660 |
| Average Phase 1 | 49.53 | Average Phase 2 | 44 |

**Table 2 Time of each participant in each phase**

Figure 3 shows six screenshots of an educational video game about the multiplication tables, which was created by a student in the contest. In Figure 3A there is an explanation of the lesson, followed by a multiplication table in Figure 3B. After watching all the multiplication tables, the student is asked several questions in which he will have to choose the correct answer (Figure 3C and Figure 3E). The creator of the game chooses to include feedback for the correct/incorrect answers (Figure 3D). Once the level is finished, the player can check his score immediately and see how many questions he has answered correctly and how many he has failed (Figure 3F).

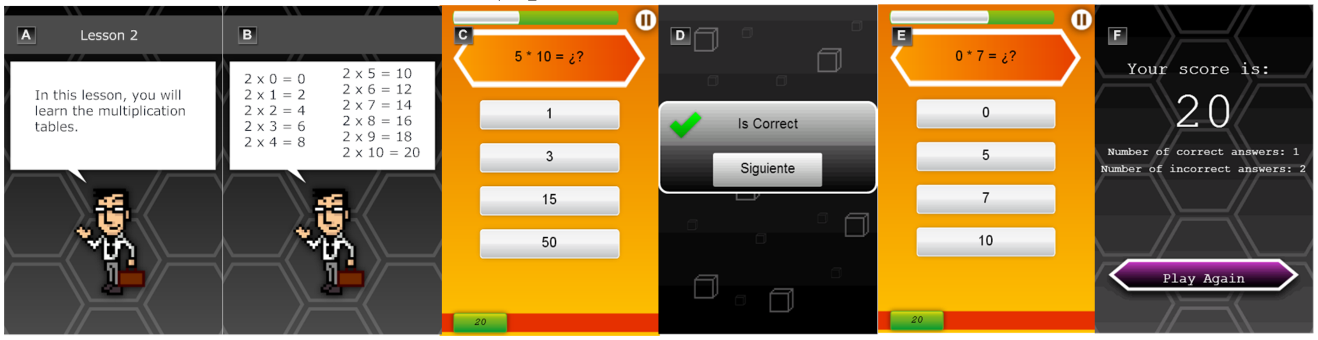


**Figure 3 Example of an educational video game**

The main conclusion of this evaluation was that, while the templates for each platform took roughly 80-125 hours of development time by expert programmers (depending on the platform), participants without without prior knowledge were able to create their multi-platform educational video game in 30-60 minutes.

## 5 Conclusions and Future Work

As we have described in this paper, we managed to build a DSL able to create multiplatform educational video games in a simple and efficient way, oriented to people without technical knowledge in the creation of video games. We implemented and tested the proposed DSL in the Gade4All editor in two occasions with satisfactory results. In our case, teachers could create multiplatform educational video games in **less than 1 hour**.

This paves the way to educational innovation models driven by enthusiastic teachers that create games for their students. In an ideal case, a teacher could create a small game in one evening based on significant events (e.g. something currently on the news) and then distribute it to the students to play in their devices (smartphones from any platform, tablets or laptops) the following day.

The goal is not to create innovative AAA games that are beyond the budget of any educational budget and require specific devices for playing them. We aim to facilitate the agile creation of low-cost educational games that are flexible and highly adaptable. This should normalize the experience of playing in the classroom, and therefore reduce the barriers for future, more complex, and expensive game development projects. While it is true that the generated educational video games might look simple for being in 2D, they can be attractive enough for this normalization of educational play.

This therefore just a small step towards the acceptance of educational video games. While this novel research can provide an easy and quick way to teachers for creating educational video games that are based on questions, future steps require going further.

First of all, further evaluations of the overall process and the game formats is required, given that the sample from this work was relatively small and focused. More games for more disciplines should be created and thoroughly tested. In addition, the evaluation focused on the agility of the creation process, rather than on the quality and applicability of the generated games, which should in turn be the object of further experiments.

Regarding the resulting games, 3D educational video games would be more aligned with student expectations. It would be desirable to create educational video games in 3D, while keeping the costs down. For example, we could include a 3D object that could be rotated by the student to observe it from any angle: a sculpture, a Rubik Cube, polyhedrons made from blueprints, etc.

More complex game models would also make the experience richer, although with a risk of making the process more complex. Fortunately, adding new options to the DSL to be able to generate new types of questions, have a bigger customisation capacity or offering more versatility would be relatively easy. It should be feasible to incrementally improve the scope of the DSL so that the approach can be used for the development of other game types. This would be necessary if we wanted to perform one of the two future Works described before.

All in all, this is a first step that aims to unlock the current situation of educational video games, where academics are increasingly certain of their potential, but the practical challenges are preventing their adoption. While these games may be simple, the agile and streamlined approach for creating multiplatform educational games can be a driving factor for their widespread adoption.

## 6 Acknowledgment

This work was performed by the ‘Ingeniería Dirigida por Modelos MDE-RG’ research group at the University of Oviedo under Contract No. MITC-11-TSI-090302-2011-11 of the research project Gade4all. Project co-financed by the Ministry of Industry, Tourism, and Commerce under its National Plan for Scientific Research, Development and Technological Innovation.

REFERENCES

[1] Fundación Telefónica, “La Sociedad de la Información en España 2014,” 2015.

[2] E. Ozcelik, N. E. Cagiltay, and N. S. Ozcelik, “The effect of uncertainty on learning in game-like environments,” *Comput. Educ.*, vol. 67, pp. 12–20, Mar. 2013.

[3] M. Papastergiou, “Exploring the potential of computer and video games for health and physical education: A literature review,” *Comput. Educ.*, vol. 53, no. 3, pp. 603–622, Nov. 2009.

[4] G. J. Hwang and P. H. Wu, “Advancements and trends in digital game-based learning research: A review of publications in selected journals from 2001 to 2010,” *Br. J. Educ. Technol.*, vol. 43, no. 1, pp. 6–10, 2012.

[5] I. Granic, A. Lobel, and R. C. . M. E. Engels, “The benefits of playing video games,” *Am. Psychol.*, vol. 69, no. 1, pp. 66–78, 2014.

[6] B. Yuan, E. Folmer, and F. C. Harris, “Game accessibility: A survey,” *Univers. Access Inf. Soc.*, vol. 10, pp. 81–100, 2011.

[7] T. M. Connolly, E. a. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, “A systematic literature review of empirical evidence on computer games and serious games,” *Comput. Educ.*, vol. 59, no. 2, pp. 661–686, 2012.

[8] B. Pourabdollahian, M. Taisch, and E. Kerga, “Serious games in manufacturing education: Evaluation of,” *Procedia Comput. Sci.*, vol. 15, pp. 256–265, 2012.

[9] A. Moshirnia, “The Educational Potential of Modified Video Games,” *Issues Informing Sci. Inf. Technol.*, vol. 4, pp. 511–521, 2007.

[10] H. Tüzün, M. Yılmaz-Soylu, T. Karakuş, Y. İnal, and G. Kızılkaya, “The effects of computer games on primary school students’ achievement and motivation in geography learning,” *Comput. Educ.*, vol. 52, no. 1, pp. 68–77, Jan. 2009.

[11] J. Torrente, Á. Blanco, E. J. Marchiori, P. Moreno-ger, B. Fernández-Manjón, and Á. Del Blanco, “Introducing Educational Games in the Learning Process,” *IEEE Educ. Eng. EDUCON 2010 Conf.*, vol. 127, no. 10, pp. 1121–1126, 2010.

[12] J. P. Gee, “Good video games and good learning,” *Phi Kappa Phi Forum*, vol. 85, no. 2, p. 33, 2005.

[13] K. Wattanatchariya, S. Chuchuaikam, and N. Dejdumrong, “An educational game for learning wind and gravity theory on iOS: Drop donuts,” in *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2011, no. Figure 1, pp. 387–392.

[14] V. Alonso Secades and O. Arranz, “Big Data and eLearning: A Binomial to the Future of the Knowledge Society,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 3, no. 6, p. 29, 2016.

[15] e-UCM research group (Complutense University of Madrid), “eAdventure,” 2006. [Online]. Available: http://e-adventure.e-ucm.es/. [Accessed: 05-Dec-2016].

[16] Lifelong Kindergarten, “Scratch,” 2015. [Online]. Available: https://scratch.mit.edu/. [Accessed: 05-Dec-2016].

[17] M. Kordaki, “Diverse Categories of Programming Learning Activities could be Performed within Scratch,” in *4th WORLD CONFERENCE ON EDUCATIONAL SCIENCES (WCES-2012) 02-05*, 2012, vol. 46, pp. 1162–1166.

[18] Alice Team, “Alice,” 2015. [Online]. Available: http://www.alice.org/. [Accessed: 05-Dec-2016].

[19] K. Johnsgard and J. McDonald, “Using Alice in overview courses to improve success rates in Programming I,” in *Software Engineering Education and Training, 2008. CSEET ’08. IEEE 21st Conference on*, 2008, pp. 129–136.

[20] R. Afreen, “Bring Your Own Device (BYOD) in Higher Education: Opportunities and Challenges,” *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 3, no. 1, pp. 233–236, 2014.

[21] P. Lavín-Mera, J. Torrente, P. Moreno-Ger, J. A. Vallejo Pinto, and B. Fernández-Manjón, “Mobile Game

Development for Multiple Devices in Education," in *International Journal of Emerging Technologies in Learning (iJET)*, 2007, vol. 4, no. s2, pp. 1–8.

[22] A. Van Deursen, P. Klint, and J. Visser, "Domain-specific languages: an annotated bibliography," *ACM Sigplan Not.*, vol. 35, no. 6, pp. 26–36, 2000.

[23] P. Moreno-Ger, D. Burgos, I. Martínez-Ortiz, J. L. Sierra, and B. Fernández-Manjón, "Educational game design for online education," *Comput. Human Behav.*, vol. 24, no. 6, pp. 2530–2540, Sep. 2008.

[24] D. R. Michael and S. L. Chen, *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade., 2005.

[25] E. J. Marchiori, J. Torrente, Á. Del Blanco, P. Moreno-Ger, P. Sancho, and B. Fernández-Manjón, "A narrative metaphor to facilitate educational game authoring," *Comput. Educ.*, vol. 58, no. 1, pp. 590–599, 2012.

[26] M. Kam, V. Rudraraju, A. Tewari, and J. Canny, "Mobile gaming with children in rural India: Contextual factors in the use of game design patterns," in *Proceedings of the 3rd Digital games research association international conference (DiGRA '07)*, 2007, pp. 24–28.

[27] C. González García, J. P. Espada, B. C. P. G-Bustelo, and J. M. Cueva Lovelle, "Swift vs. Objective-C: A New Programming Language," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 3, no. 3, pp. 74–81, 2015.

[28] J. Torrente, B. Borro-Escribano, M. Freire, Á. Del Blanco, E. J. Marchiori, I. Martínez-Ortiz, P. Moreno-Ger, and B. Fernández-Manjón, "Development of game-like simulations for procedural knowledge in healthcare education," *IEEE Trans. Learn. Technol.*, vol. 7, no. 1, pp. 69–82, 2014.

[29] Federation of American Scientists, "Harnessing the power of video games for learning," *Summit on Eucational Games*, Washington, p. 53, 2005.

[30] M. Griffiths, "The educational benefits of videogames," *Educ. Heal.*, vol. 20, no. 3, pp. 47–51, 2002.

[31] J. P. Gee, "What video games have to teach us about learning and literacy," *Comput. Entertain.*, vol. 1, no. 1, p. 20, Oct. 2003.

[32] A. L. Brown, "The advancement of learning," *Educ. Res.*, vol. 23, no. 8, pp. 4–12, 1994.

[33] M. Graafland, J. M. Schraagen, and M. P. Schijven, "Systematic review of serious games for medical education and surgical skills training," *Br. J. Surg.*, vol. 99, pp. 1322–1330, 2012.

[34] E. A. Akl, K. M. Sackett, W. S. Erdley, R. A. Mustafa, M. Fiander, C. Gabriel, and H. Schuenemann, "Educational games for health professionals," *Cochrane Database Syst. Rev.*, no. 3, p. CD006411, 2013.

[35] R. Kanthan and J. L. Senger, "The impact of specially designed digital games-based learning in undergraduate pathology and medical education," *Arch. Pathol. Lab. Med.*, vol. 135, pp. 135–142, 2011.

[36] M. Boeker, P. Andel, W. Vach, and A. Frankenschmidt, "Game-based e-learning is more effective than a conventional instructional method: A randomized controlled trial with third-year medical students," *PLoS One*, vol. 8, no. 12, pp. 1–11, 2013.

[37] M. T. Cheng, T. Su, W. Y. Huang, and J. H. Chen, "An educational game for learning human immunology: What do students learn and how do they perceive?," *Br. J. Educ. Technol.*, vol. 45, no. 5, pp. 820–833, 2013.

[38] M. Heron, V. L. Hanson, and I. W. Ricketts, "Accessibility Support for Older Adults with the ACCESS Framework," *Int. J. Hum. Comput. Interact.*, vol. 29, pp. 702–716, 2013.

[39] T. Westin, K. Bierre, D. Gramenos, and M. Hinn, "Advances in Game Accessibility from 2005 to 2010," in *Proceedings of the 6th International Conference on Universal Access in Human-computer Interaction: Users Diversity - Volume Part II*, 2011, pp. 400–409.

[40] S. Cano, D. M. Alghazzawi, J. M. Arteaga, H. M. Fardoun, C. A. Collazos, and V. B. Amador, “Applying the information search process model to analyze aspects in the design of serious games for children with hearing impairment,” *Univers. Access Inf. Soc.*, vol. 17, no. 1, pp. 83–95, Mar. 2018.

[41] S. Kent, “Model Driven Engineering,” *Comput. Comput. Soc.*, vol. 2335, no. 2, pp. 286–298, 2002.

[42] E. W. Dijkstra, “The humble programmer,” *Commun. ACM*, vol. 15, no. October 1972, pp. 859–866, 1972.

[43] E. Palacios-González, H. Fernández-Fernández, V. García-Díaz, B. C. P. G-Bustelo, J. M. C. Lovelle, and O. S. Martínez, “General purpose MDE tools,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 1, no. 1, pp. 72–75, 2008.

[44] E. M. Schön, M. Escalona, and J. Thomaschewski, “Agile Values and Their Implementation in Practice,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 3, no. 5, p. 61, 2015.

[45] V. García-Díaz, H. Fernández-Fernández, E. Palacios-González, B. C. P. G-Bustelo, O. Sanjuán-Martínez, and J. M. C. Lovelle, “TALISMAN MDE: Mixing MDE principles,” *J. Syst. Softw.*, vol. 83, no. 7, pp. 1179–1191, Jul. 2010.

[46] V. García-Díaz, J. B. Tolosa, B. C. P. G-Bustelo, E. Palacios-González, O. Sanjuán-Martínez, and R. G. Crespo, “TALISMAN MDE Framework: An Architecture for Intelligent Model-Driven Engineering,” in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, no. November, S. Omatu, M. P. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, and J. M. Corchado, Eds. Springer Berlin Heidelberg, 2009, pp. 299–306.

[47] J. Fabra, V. De Castro, P. Álvarez, and E. Marcos, “Automatic execution of business process models: Exploiting the benefits of Model-driven Engineering approaches,” *J. Syst. Softw.*, vol. 85, no. 3, pp. 607–625, Mar. 2012.

[48] C. González García, C. P. García-Bustelo, J. P. Espada, and G. Cueva-Fernandez, “Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios,” *Comput. Networks*, vol. 64, no. C, pp. 143–158, Feb. 2014.

[49] C. González García, J. P. Espada, E. R. N. Valdez, and V. García-Díaz, “Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform,” in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 352–357.

[50] B. Selic, “MDA manifestations,” *Eur. J. Informatics Prof.*, vol. IX, no. 2, pp. 12–16, 2008.

[51] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, “Relevance, benefits, and problems of software modelling and model driven techniques—A survey in the Italian industry,” *J. Syst. Softw.*, vol. 86, no. 8, pp. 2110–2126, Aug. 2013.

[52] A. van Deursen and P. Klint, “Little languages: Little maintenance?,” *J. Softw. Maint.*, vol. 10, no. 2, pp. 75–92, Apr. 1998.

[53] D. Hästbacka, T. Vepsäläinen, and S. Kuikka, “Model-driven development of industrial process control applications,” *J. Syst. Softw.*, vol. 84, no. 7, pp. 1100–1113, Jul. 2011.

[54] A. Van Deursen, “Domain-Specific Languages versus Object-Oriented Frameworks: A financial engineering case study,” *Smalltalk Java Ind. Acad.*, pp. 35–39, 1997.

[55] V. García-Díaz, J. Pascual-Espada, C. Pelayo G-Bustelo, and J. M. Cueva-Lovelle, “Towards a Standard-based Domain-specific Platform to Solve Machine Learning-based Problems,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 3, no. 5, p. 6, 2015.

[56] P. Moreno-Ger, I. Martínez-Ortiz, and B. Fernández-Manjón, “The <E-Game> Project: Facilitating the Development of Educational Adventure Games,” in *International Association for*

*Development of the Information Society (IADIS) Cognition and Exploratory Learning in Digital Age (CELDA)*, 2005, pp. 353–358.

[57] P. Lavín-Mera, P. Moreno-Ger, and B. Fernández-Manjón, “Development of Educational Videogames in m-Learning Contexts,” *2008 Second IEEE Int. Conf. Digit. Game Intell. Toy Enhanc. Learn.*, pp. 44–51, 2008.

[58] A. Serrano, E. J. Marchiori, A. del Blanco, J. Torrente, and B. Fernandez-Manjon, “A framework to improve evaluation in educational games,” in *Proceedings of the 2012 IEEE Global Engineering Education Conference (EDUCON)*, 2012, pp. 1–8.

[59] YoYoGames, “GameMaker Studio,” 2013. [Online]. Available: http://www.yoyogames.com/gamemaker/studio. [Accessed: 05-Dec-2016].

[60] A. Baytak and S. M. Land, “A case study of educational game design by kids and for kids,” *Procedia - Soc. Behav. Sci.*, vol. 2, no. 2, pp. 5242–5246, Jan. 2010.

[61] M. Duggan, *Making a GameSalad for Teens*. Cengage Learning PTR, 2013.

[62] K. Roy, W. C. Rousse, and D. B. DeMeritt, “Comparing the mobile novice programming environments: App Inventor for Android vs. GameSalad,” *2012 Front. Educ. Conf. Proc.*, pp. 1–6, Oct. 2012.

[63] Unity Technologies, “Unity,” 2013. [Online]. Available: http://unity3d.com/. [Accessed: 05-Dec-2016].

[64] W. A. Mattingly, D. Chang, R. Paris, N. Smith, J. Blevins, and M. Ouyang, “Robot design using Unity for computer games and robotic simulations,” *2012 17th Int. Conf. Comput. Games*, pp. 56–59, Jul. 2012.

[65] E. R. Núñez-Valdez, O. Sanjuán-Martínez, B. C. P. G-Bustelo, J. M. C. Lovelle, and G. Infante-Hernandez, “Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 2, no. Regular Issue, pp. 33–42, 2013.

[66] E. R. Núñez-Valdez, V. García-Díaz, J. M. C. Lovelle, Y. S. Achaerandio, and R. González-Crespo, “A model-driven approach to generate and deploy videogames on multiple platforms,” *J. Ambient Intell. Humaniz. Comput.*, vol. 8, no. 3, pp. 435–447, Jun. 2017.

[67] Eclipse, “Ecore,” 2010. [Online]. Available: http://wiki.eclipse.org/Ecore. [Accessed: 28-Sep-2016].