Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering

Edward Rolando Nuñez-Valdez[1], Oscar Sanjuan[2], Begoña Cristina Pelayo Garcia-Bustelo[1], Juan Manuel Cueva-Lovelle[1], Guillermo Infante Hernandez[1]
[1]*University of Oviedo, Computer Science Department, Sciences Building, Oviedo, Spain*
[2] *University Carlos III of Madrid, Computer Science and Engineering Department, Madrid, Spain*.

***Abstract* — The development of applications for mobile devices is a constantly growing market which and more and more enterprises support the development of applications for this kind of devices. In that sense, videogames for mobile devices have become very popular worldwide and are now part of highly profitable and competitive industry. Due to the diversity of platforms and mobile devices and the complexity of this kind of applications, the development time and the number of errors within that development process have increased. The productivity of the developers has also decreased due to the necessity of using many programming languages in the development process. One of the most popular strategies is to employ specialized people to perform the development tasks more efficiently, but this involves an increase of the costs, which makes some applications economically unviable. In this article we present the Gade4all Project, consisting in a new platform that aims to facilitate the development of videogames and entertainment software through the use of Domain Specific Languages and Model Driven Engineering. This tool makes possible for users without previous knowledge in the field of software development to create 2D videogames for multiplatform mobile devices in a simple and innovative way.**

***Keywords* — DSL, Gade4all , MDE, VideoGame**

I. INTRODUCTION

The use of new technologies (like smartphones, tablets, computers and the Web) as tools for the use of videogames and entertainment applications is quite extended nowadays. This expansion has lead to the appearance of a great diversity of platforms and mobile devices that allow to execute a wide range of applications.

The existance of different types of platforms and mobile operative systems and the constant evolution and appearance of new devices has brought with it an increase of the development time and the appearance of errors during that process. Also, the productivity of the developers of mobile applications has decreased, generally due to the necessity of developing the same application using many programming languages in order to publish it in other platforms and reach a higher number of users in the market.

Also, the development of games is generally aimed to people with extensive knowledge in the development of this kind of applications. Normally, an important part of these applications are developed for mobile devices and the Web [1].

All this suggests that the development of videogames can be much more profitable and optimized if we can abstract the main characteristics of many videogame typologies. This abstraction serves a base for the reuse of code and the generation of multiplatform videogames in an automatic way.

With the objective of minimize the aforementioned inconveniences, improve the efficiency and solve the typical problems of software development [2][3], this project proposes the use of Model Driven Engineering (MDE) [4] to develop a tool that allows to create multiplatform videogames in a quick and simple way. This tool can be used by anyone, even if that person lacks programming knowledge.

By using typologies templates and the most advanced technology for automated generation of code based on Domain Specific Languages and Model Driven Engineering, this tool allows the development of 2D multiplatform games. These applications are automatically generated in native code for platforms like Windows Phone, Android, iOs and HTML5, managing to optimize the development time, raise the efficiency and reduce the number of errors.

The reminder of this paper is structured as follows: in Section 2 we present a summary of the current problems that affect the development of videogames. Section 3 presents the background of the videogames, some videogame development tools and the general concepts of Model Driven Engineering and Domain Specific Language. Section 4 shows the gade4all case of study, Section 5 contains the conclusion and future research directions of this work and Section 6 shows the Acknowledgement.

II. PROBLEMS

In their beginning, videogames were developed by very small groups of people and they could only be executed by a small amount of devices, some of which had a very high price. As the videogame industry evolved, so did the size of the development teams and the variety of devices able to execute

DOI: 10.9781/ijimai.2013.224

those videogames. This is because the growing popularity of mobile phones has encouraged software development enterprises to produce a greater number of videogame applications.

Roughly speakning, we could conclude that the development process of multiplatform videogames is facing the following problems:

A. Complexity within videogame development

Due to the existing multiple mobile platforms for the development and deployment of videogames, the development of this kind of applications has become harder. This is because it's necessary of developers to have an extensive knowledge of the videogame's domain, as well as experience in the development of multi-device software.

In that sense, the development of videogames requires not only experience in the development of software, but also a great knowledge of the different programming languages that are used to create multiplatform videogames. This increases the complexity of videogame development and also the necessity of having a highly qualified person that can handle the problem.

B. Huge variety of platforms and technologies

Due to the existence of a high diversity of languages and platforms like Java, Objetive C, Microsoft .NET, HTML5 and others that allow to develop applications for their deployment on devices based on Android, iOS, Window or the different web browsers. This often causes the development of applications for specific technologies to be poorly feasible. This is because each technology fills a different market niche, which reduces the number of potential user for the developed product. Fort the enterprises focused in videogame development, this implies having development teams for each platform. This way, they can make sure that their products will be well positioned in the different market niches and make them more feasable.

C. Increase of development time and implementation costs

This is a problema that arises from the complexity of the development of multiplatform and multi-device software. Due to the great diversity of platforms and devices, it's often necessary to develop the same application for different devices and operative systems. This causes a reduction of the speed and effectivity of the software development process, specially when developing multiplatform products. Currently, if we want to generate native applications for the different existing platforms, we have to develop the same application with different programming languages and in different environments, which considerably raises the amount of time and money necessary for the development of the application.

III. BACKGROUND

Videogames are applications developed for personal entertainment that are based on the interaction of one or more persons with the electronic device in which its being executed. In most cases, videogames represent virtual environments in which the player can control one or more characters with the objective of fulfilling objectives according to the rules of the game.

Since the 70s, videogames have become popular worldwide and are now a highly profitable and competitive industry. The "trend" of videogames has been kept afloat thanks to the great diversity they offer, as well as the evolution of their technology and creativity. Originally, videogames were created and designed to be executed in computational systems, but in the middle 70s the first home video game consoles appeared and became quite popular in a short time, hitting 13 millions of units only in the United States [5].

The devices and the way of playing have also experienced a big evolution, from the buttons and sticks of the old games to the motions sensors and tactile interfaces of today. This is possible thanks to the technological advances and the inventiveness of the creators of the games, which take advantage of the available resources to expand the possibilities. The revolution of the Web and the mobile devices represents a new era for videogames, as there are millions of them in the Web prepared to be executed in a web browser and many mobile devices and platforms, like Android, iOS, Windows Phone and others.

To facilitate the creation of videogames for all these platforms and devices, several graphic tools have been developed to help videogame creators and users in general to generate videogames in an easy way. This section describes some of the videogame editors that help to perform this task. Also, it shows the general concepts about Model Driven Engineering and Domain Specific Language as software development approaches, which served as a base for the development of the Gade4all platform.

A. Videogame Editors

Nowadays there are many videogame editors that make easier to develop this kind of applications. Some of these tools allow to export videogames to different platforms like Windows, iOS, Android, Flash and HTML5. The main differences between these tools are the supported platforms, the typologies of games that can be generated, the degree of customization of the characteristics and behaviours of the elements present in the games and, most importantly, their simplicity and accessibility. Below are some examples of videogame editors that can be found in the market:

1) GameSalad

It's a tool that allows to develop videogames and export them for its distribution in different devices and platforms: Macs, PCs, iOS smart devices, Android, and HTML5 [6]. The development of games with GameSalad Creator is performed through scenes and actors, whose attributes can be defined to stablish a behavior, physical appearance, sounds and rules. However, while it has many advantages that make easy to develop them, the videogames created with GameSalad can

only be edited from this tool, which doesn't make possible for other developers to expand them using the native development environments.

2) GameMaker Studio

GameMaker Studio is a tool for the creation of videogames, aimed both to novice users and experimented professionals of videogame development. It's quite intuitive, because it was originally designed for inexperienced users to learn the basic concepts of programming, understand game's architecture and create games in a simple way [7]. In [8] it's shown how some children designed educative videogames that demonstrated their understanding of the GameMaker software. This tool is based in its own interpreted language called GML (Game Maker Language), which is based in scripts aimed to expert users. The program is regulated by the Drag and Drop philoshopy, so supposedly is not necessary to write a single line of code. It allows the generation of videogames for HTML5 and the platforms iOS, Android, Windows.

3) Stencyl

It's a tool for the creation of videogames developed by StencylWorks [9]. This product offers videogame designers a graphical editor that allows them to create games and export them to iOS, Mac, Windows and Flash formats. Since version 3.0 (version beta) Stencyl allows to export applications to Android and HTML5. The creation of games in Stencyl is based in the definition of scenes and actors. Each scene corresponds to a level in the game, and the different actors are added to those scenes. An actor is any element that is shown in the level, from the graphical elements without added logic or functionality to the characters of the game, with their own rules and behaviors. The development environment of Stencyl is intuitive enough for people with experience in design program, and though it has several default templates, it demands programming and algorithmic logic knowledge, as many of the game's modifications require the creation of algorithm flowcharts.

B. Model Driven Engineering

MDE, or "Model Driven Engineering" is an approach that suggests the use of models as the main axis of the lifecycle of a software and saves time and effort in the development process [10]. Those models are abstractions that are focused in the development of specific applications of a problem.

The implementation of Model Driven Engineering allows the raise of productivity through the reutilization of standardized models, thus simplifying the development process through recurring templates of an specific domain, and, at the same time, creating an standardization of the terminology and the best practices used in the application of the domain.

Model Driven Engineering possesses a wide spotlight architecture and automation with high levels of abstraction in software development. This abstraction promotes the design of simple models with a wide perspective for the resolution of specific problems, which combined with the adequate semantic, raise the total rate of possible automation. The OMG (Object Management Group) has developed a set of standards referred to as MDA (Model Driven Architecture).

According to [10] a model is the description of one or more elements of the domain or real world. There is also the concept of metamodel, which presents the tools that allow the creation a model through its description. Metamodels also describe themselves, generating an extremely high abstraction in which all the models match each other.

The use of Model Driven Engineering is key for the development of this project, as the concepts about the definition of models with high levels of abstraction efficiently adjusts to the developed tool. By applying the principles of this engineering we manage to abstract the main characteristics of the supported videogames, making easier to create a Domain Specific Language that allows the generation of multiplatform games in a simple and easy way.

C. Domain Specific Language

DSL, or "Domain Specific Language" is a usually declarative language that focus on the resolution of problems of particular domain. In many cases, these solutions can be translated to library calls or common subroutines, in which the DSL can be seen as a way of occulting the details of that library [11]

As [12] describe, models can be clearly expressed by using a DSL whose language is specifically designed to meet the specific needs of the application's domain. In that sense, the DSL contains specific concepts of the domain that allow a better understanding of it and ignoring elements that moves away from the reality of the domain.

As **Fig.8** shows, one of the main characteristics of the DSL is that it allows the implementation of a definition that is independent from the platform and, once finished, will go through a transformation process, offering the construction of a proyect in a specific platform as a final result. The DSL defined for the Gade4all project makes easier to generate complete applications for the iPhone, Android, Windows Mobile and HTML5 platforms.

Fig.8. Transformation of platform independent models

DSL is crucial for Model Driven Engineering and so for the project. The DSL used for the definition of the videogames contains a high level of abstraction that allows to define all the elements that are necessary for the generation of a videogame, as well as the stablishment of the required values to assign a behavior to the actors. That task is performed through a videogame editor, so the final user doesn't need to know its details.

IV. CASE STUDY

In the last years several operative systems for mobile devices have appeared, among which we can mention: iOS, developed by Apple, Windows Phone, developed by Microsoft and Android, developed by Google. The birth of the Smartphone has unleashed a war in the development of applications for mobile devices based on the aforementioned operative systems.

The necessity of developing applications that can be executed in different platforms has brought with it the appearance of problems that were described in Section II. In order to solve those problems, it's necessary to create a tool that allows to develop applications for different platforms in an agile, efficient and easy way.

In the context of videogames, a tool that allows to reduce the complexity of the developing of multiplatform videogames will also allow to minimize the costs and development time, as well as reducing the appearance of errors in this kind of application.

In this case study we show the Gade4all videogame platform, which has been developed with the purpose of making possible for a wide range of inexperienced users to create multiplatform videogames in an easy way. Because many videogames share a series of characteristics and behavior patterns, Gade4all has defined a high level language that allows the creation of a great variety of videogames in an easy and effortless way, all through the abstraction of the most relevant characteristics of the main videogame typologies.

A. Gade4all

Gade4all is an innovative project that aims to simplify the development of multiplatform videogames for mobile devices and the Web. Through the use of a graphical tool, it allows inexperienced users to develop software and create innovative videogames in a quick and simple way. The Gade4all tool, which has been developed using Model Driven Engineering as a base, facilitates the development of videogames in a dynamic way by using a Domain Specific Language (DSL). This platform allows the definition of the elements, characteristics and rules of videogames in an easy way. The use of this editor as a tool for the development of videogames makes easier to get in touch with the most demanded technologies and devices nowadays, like: Android, iPhone, Windows Phone and HTML5.

This tool allows an agile development of 2D videogames that are automatically generated in native code for multiple platforms, allowing the improvement of the development time, the reduction of the number of errors and the raising of productivity.

The generation of applications in native code offers different advantages, among which we can highlight the real use of the functionalities of each platform and the possible incorporation of improvements by developers with knowledge of the environment. In that sense, the generated games can be used as prototypes for more complex applications.

Besides, the platform is oriented to create videogames and entertainment software of any kind, quickly enough to create them as support applications to advertise recent events, reducing costs and development time.

On its first stage of development, this platform supports a series of videogame typologies like: touch, trivial, strategy, puzzle and platform. These make easier to create videogames for devices like Android, iPhone, Windows Phone y web navigators that support HTML5.

B. Gade4all platform Archytecture

As shown in Figure 2, through the use of a graphical editor, the Gade4all platform allows the creation of videogames easily and intuitively. This allows us to create all the elements, actors and rules of the game and offers all the necessary mechanisms for the configuration and use of the DSL that describes the model of videogames automatically and dynamically. Once the application has been designed with the editor, an integrated transformation engine uses the DSL and the default templates to generate a complete application in native code, which will be ready to be executed in the selected platform. Below we describe the different elements that the Gade4all platform's architecture is composed of:

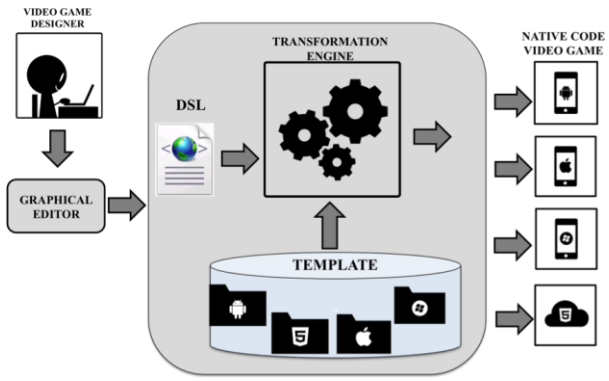

Fig. 9. Gade4all Platform Arquitecture

1) Graphical editor

Many videogames share several patterns of behavior and we intend to take some of the most representative to allow the definition of videogames in a quick and simple way, just by modifying some of the properties of those behaviors.

In order to do this, it's necessary to define the variable elements that make a videogame (graphic aspects, sounds, characters…) and specially their behavior of those elements, so we can stablish the rules of the game.

The graphical editor allows both videogame designers and inexperienced users to define the characteristics and rules of the game through a graphical interface that makes the development of videogames more intuitive.

If a user wants to define a videogame using this tool, he will only have to follow some interactive steps, through user-friendly screens that make the process easier and quicker. This sequence starts with the selection of a typology and ends with the selection of the platform in which we want to deploy the videogame. **Fig. 10** and figure**Fig. 11** show the graphical

interfaces of the videogame typology selection screen and the deployment platform selection screen, respectively.

Fig. 10. Interface for the selection of the videogame typology

Fig. 11. Interface for the selection of the deployment platform

The steps to follow can differ from one another, as some of the characteristics and behaviors of the games are different. For instance, a trivial game has different characteristics than a platform game.

Fig. 12 shows the steps that a user must follow to define a platform game, each of one its composed by user-friendly screens that make easier to define the characteristics and behaviors of the videogame even if the user does not possess technical knowledge at all.

Some of these screens represent the pre-configured main elements, which can be used with their default values or can be modified by the use according to the characteristics he wants to include in its videogame.

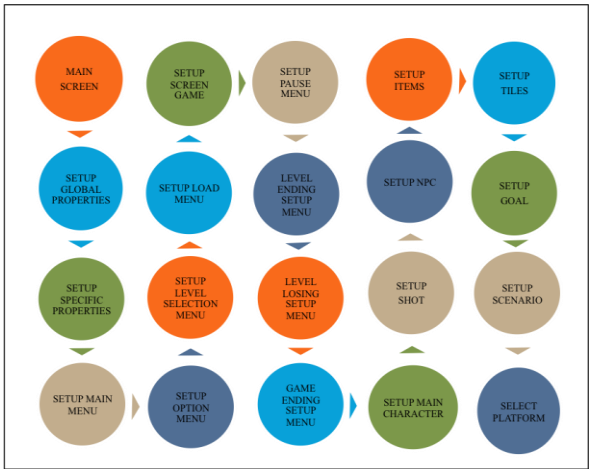

Fig. 12. Screen structure for the creation of a platform game

Fig. 13 shows a graphical interface that allows the user to create a library of characters in a platform videogame. Here is where the main characteristics of the characters are defined: size of the character, number of lives, sounds, animations, etc. The characters defined in this screen are the ones that will be used in the defined scenes defined for the levels of the videogame.

The rest of the screens defined in the editor allow the user to configure the different components that will appear in the videogame: menus (main menu, options menu, pause menu, game, etc), elements (main character, NPCs, obstacles, items, tiles,etc) and scenes (the levels of the videogame). These screens have been defined following the same structure, so the users can interact with them intuitively.

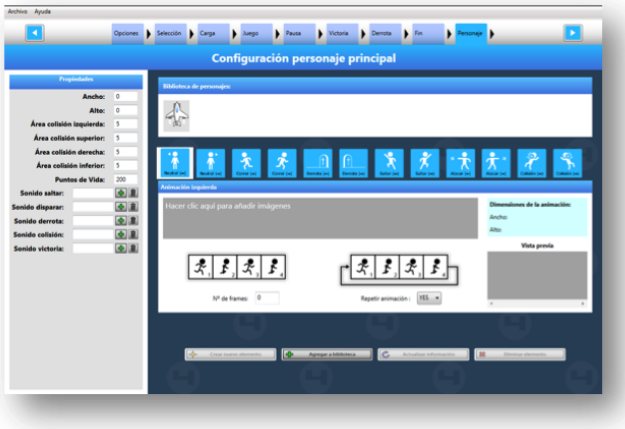

Fig. 13. Interface for the creation of the main character in a platform game

2) Domain Specific Language

On of the main pillars of this platform is the domain specific language that defines the main characteristics of the different videogame typologies supported by the platform. Thanks to the graphical editor, this high level language defined in XML allows videogame designers to define the rules of the game they are developing in a simple way. This language makes easier for inexperienced users to develop videogames in a very intuitive way.

Fig. 14 shows a fragment of DSL corresponding to the "Properties" section of the screen shown in Figure 6. These values are modified by the user from the graphical editor. Generally, the creation process will result in the obtention of a XML file containing all the defined elements corresponding to the DSL of the developed videogame.

```
<character>
    ...
    <character_width>90</character_width>
    <character_height>90</character_height>
    <character_collision_area_left>5</character_collision_area_left>
    <character_collision_area_top>5</character_collision_area_top>
    <character_collision_area_right>5</character_collision_area_right>
    <character_collision_area_down>5</character_collision_area_down>
    <character_health>200</character_health>
    <character_sound_on_jump_source>sound1.mp3</character_sound_on_jump_source>
    <character_sound_on_fire_source>sound2.mp3</character_sound_on_fire_source>
    <character_sound_die_source>sound3.mp3</character_sound_die_source>
    <character_sound_collision_bad_item_source>sound4.mp3</character_sound_collision_bad_item_source>
    <character_sound_collision_good_item_source>sound5.mp3</character_sound_collision_good_item_source>
    ...
    ...
</character>
```

Fig. 14. Fragment of DSL from a platform game

3) Template

To generate native code, we employ a series of templates that abstract a set of common characteristics that can be used as a generic model. This templates are developed as a base for the different platforms that Gade4all supports: iPhone, Android, Windows Phone and HTML5. These projects (Template) abstract a set of attributes and generic methods that make easier to convert and import the characteristics tha result from the implementation of a request from the DSL. **Fig. 15** shows a section of the template where the "Start" button of a videogame is configured. The values of the variables wil be replaced by the transformation engine when the game is generated for the selected platform. This mechanism for the replacement of variables is commonly used on those elements that don't change their behavior too much from game to game, like option menus, stage select menus, etc.

Fig. 16 shows a section of the template where the generation and load processes of the levels of the videogame are performed. This is done with the transformation engine, using SAX (Simple API for XML) technology and Reflection.

This dynamic loading mechanism is used when the behavior of the videogame's elements changes considerably depending on the levels and the typologies. For instance, the positioning and behaviors of the characters within the scene).

Also, because the scene does not know how many elements or characters will appear in every level, we need a a mechanism like this, which offers flexibility to load the elements required for the definition and configuration of a videogame.

```
public int imageButtonStart = %main_screen_start_button_image_source%;
public int heightImageButtonStart = %main_screen_start_button_image_height%;
public int widthtImageButtonStart = %main_screen_start_button_image_width%;
public int posXButtonStartMenu = %main_screen_start_button_x_position%;
public int posYButtonStartMenu = %main_screen_start_button_y_position%;
```

Fig. 15. Template design for the creation of a button

```
tiles[levelLoaded.character_x_position / Tile.Width][ levelLoaded.character_y_position / Tile.Height] =
        LoadStartTile
        (
            levelLoaded.character_width,
            levelLoaded.character_height,
            levelLoaded.character_collision_area_left,
            levelLoaded.character_collision_area_top,
            levelLoaded.character_collision_area_right,
            levelLoaded.character_collision_area_down,
            levelLoaded.character_health,
            levelLoaded.character_sound_on_jump_source,
            levelLoaded.character_sound_on_fire_source,
            levelLoaded.character_sound_die_source,
            levelLoaded.character_sound_collision_bad_item_source,
            levelLoaded.character_sound_collision_good_item_source
            ...
            ...
        );
```

Fig. 16. Template that enables the automatic loading of levels

4) Transformation Engine

The videogames' specifications performed with the DSL are processed by a transformation engine. This engine automatically generates applications in native code, which can be executed in the platforms selected by the videogame designer during the development process. The transformation engine applies a set of algorithms on the specification of the game to convert this "abstract" code into specific code (Java, C#, Objective C, HTML5) that can be executed in a mobile platform or the Web.

This transformation engine allows us to perform the following tasks in the videogame's generation process:

- Copy image and sound resources used in the editor.
- Copy of the files of each level, obtained from the editor, in order to simplify the transformation process.
- Replacement of the values of the DSL's request in the selected templates.
- Automatic loading of the different XML levels configured in the developed videogame.

Fig. 17 shows the process of automatic replacement of values in the variables defined in the template where the main characteristics of a button are abstracted.

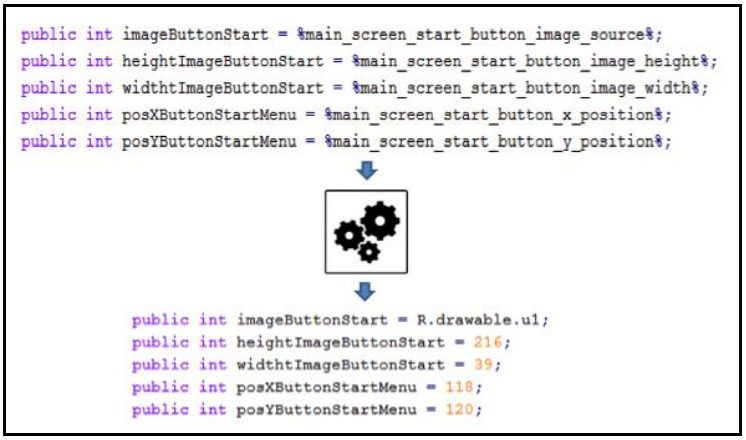

Fig. 17. Transformation process of a button's characteristics

Finally, once the transformation engine has performed all the processes for the generation of native code, the resulting projects can be imported, modified and compiled in the development environment of the selected platform: Eclipse, Xcode, Visual Studio .Net, etc.

Fig. 18 shows an example of the transformation and compilation process of a videogame developed for the

Android platform, which will generate an application that can be executed in any device that supports this type of application.

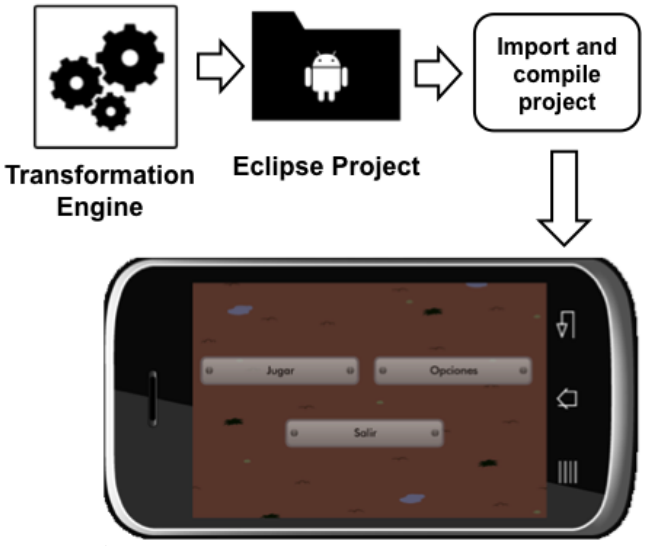

Fig. 18. Transformation and compilation process in Android

C. Supported typologies

Gade4all makes easier to develop videogames for different typologies in a flexible, user-friendly way. These typologies allow users to develop any kind of 2D videogame that matches the supported typologies. To simplify the development process and group the different types of games within a specific classification, the tool allows the development of videogames belonging to the following typologies:

a) Touch ability games

In this kind of typology we move the main character by touching the surface of the mobile phone's screen. As it can be seen in Figure 12 the mechanic of the typology allows users to create videogames based in dodging enemies and gathering items. The objectives of the level can be different: advance a certain amount of distance without crashing into enemies or obstacles, gather a special item, obtain a certain amount of points, etc.

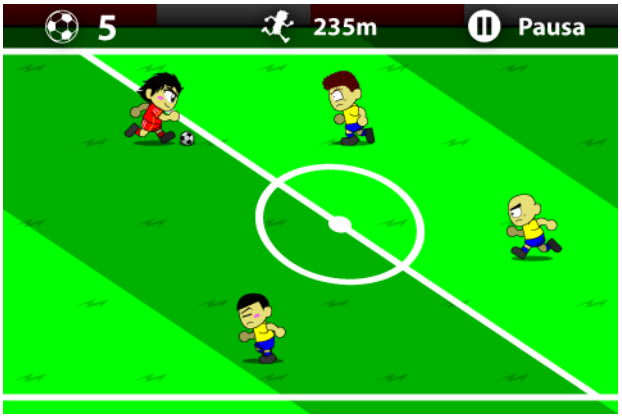

Fig. 19. Ability Touch game developed with the Gade4all tool

b) Trivial Typology

With this typology users can create many types of videogames based on questions that can contain different types of resources: texts, images, sounds, videos, etc. Each question can have one or more valid answers and even a order for the correct answers can be stablished. This typology offers lots of possibilities regarding the kind of questions: memory questions, logic questions, trivial-like questions, questions based on observation, etc.

With the support of this typology, we can generate multiplatform educative videogames to help and motivate students and teachers to take advantage of the new technologies in order to improve the process of learning.

Fig. 20 shows some examples of the type of questions that can be created with this typology. These can contribute to interest teachers in this tool and help them creatings educative videogames as a learning resource.

These resources can include tutorials, math exercises, trivials or mental agility tests to help the students to improve their learning through videogames and entertainment applications.

Fig. 20. Trivial game developed with the Gade4all tool

c) Puzzle Typology

As shown in Figure 14, with this typology we can design games in which we can connect pieces or throwing them to others. The mechanics of this typology allow users to create games in which they control the pieces that appear on the screen in order to make them collide with others. There are many possible winning conditions for each level: eliminate all the pieces of the same color, destroy all the pieces of the same color, hit a certain piece, etc.

Gade4all offers a flexibility that makes possible to develop any type of videogames based on this typology, which allows the creation of a wide range of puzzle videogames, in a simple and effortless way.

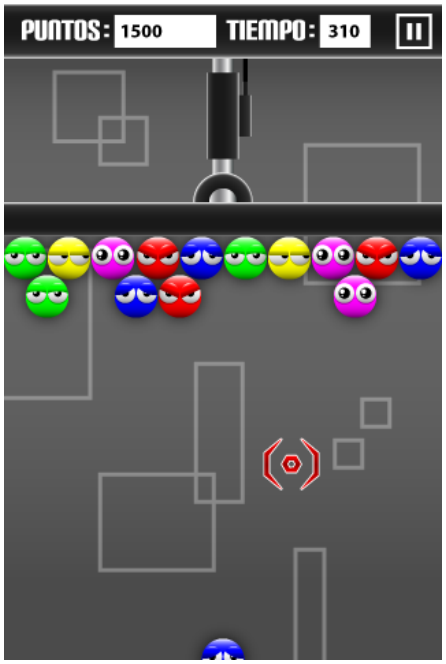

Fig. 21. Puzzle videogame developed with the Gade4all tool

d) Plataform Typology

In this typology we move the main character by using the controls that appear on the screen: horizontal movement, jump and shoot. As Figure 15 shows, the main character advances through a stage composed of blocks of different types, defeating enemies and dodging their shoots. The character can gather items in order to obtain points and the objective o each level is to reach a certain point of the stage or collecting one special item.

Fig. 22. Platform videogame developed with the Gade4all tool

e) Strategy Videogame

This typology allows user to design classic turn-based strategy videogames. As it can be seen in **Fig. 23**, in this type of videogame we define the aspect of the troops from two armies: the player's army and the computer's. The game can be composed by several levels, with different thematic, terrains and obstacles. The gameplay is turn-based, alternating between the movements of the player and the computer's. There are many possible winning conditions: defeating the leader of the other army, eliminating the whole enemy's army, capturing a flag, etc.

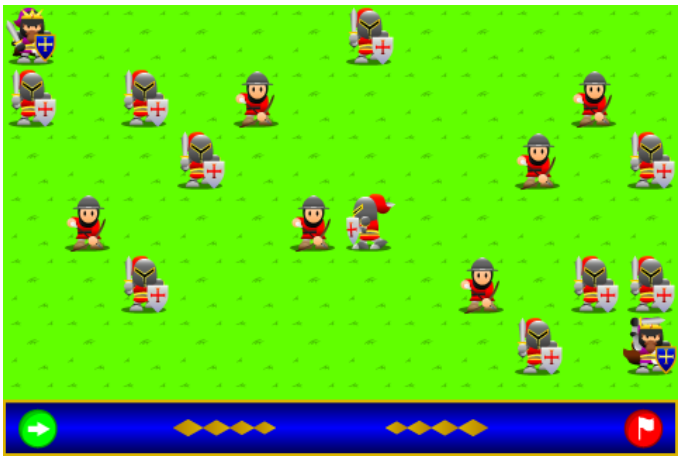
Fig. 23. Strategy videogame developed with the Gade4all tool

D. Comparison of Videogame Editors

In this section we analyze some of the characteristics of the different videogame creation tools that were previously described in this proposal. **¡Error! No se encuentra el origen de la referencia.** shows a general vision of the main similarities and differences between Gade4all and the rest of tools.

It should be noted that, as it can be seen in Table 1, the Gade4all platform offers some significant advantages in comparison to the other tools that were analyzed, among which we can highlight the fact that only Gade4all offers the possibility of developing videogames without requiring knowledge on software development. This makes possible for the tool to be used almost by more people. Also, it is the only tool that allows the complete generation of a videogame in native code for the different platforms, which makes possible for developers to modify or expand previously generated projects and create more complex applications.

1) Main similarities between Gade4all and the rest of videogame editors

After analyzing a set of tools and performing a comparison of these with the Gade4all platform, we can highlight some of the similarities that have been found:

- Possibility of creating actors and characters.
- Definition of sounds for events.
- Graphical editor for the creation and edition of levels.
- Generation of games for different platforms.
- Storage of the characteristics of a videogame for its later modification.

TABLE 1: COMPARISON OF THE MAIN TOOLS OF THE MAIN VIDEOGAME EDITOR

	GameSalad	GameMaker Studio	App Inventor	Stencyl	Gade4all
Previous knowledge on software development	Yes	Yes	Yes	Yes	No
Graphical editor of menus	No	No	Yes	No	Yes
Graphical editor of characters and levels	Yes	Yes	Yes	Yes	Yes
Definition of sounds for events	Yes	Yes	Yes	Yes	Yes
Generation of the complete game in native code, aside from the executable file	No	No	No	No	Yes
Possibility of saving the state of the game	Yes	Yes	Yes	Yes	Yes
Possibility of loading an existing game	Yes	Yes	Yes	Yes	Yes
Editing solutions outside the tool	No	No	No	No	Yes
Modifying levels outside the tool	No	No	No	No	Yes
Checking functioning of a generated level	Yes	Yes	Yes	Yes	Coming soon

S

2) Main differences between Gade4all and the rest of videogame editors.

As it has been specified in previous sections, even though the existing videogame editors share some similarities with the Gade4all tool, there are some important differences that should be mentioned:

- With Gade4all, we can configure the different menus of the game, while with the others (except App Inventor) we can only configure the game screen and its elements.
- The final result of a game created with any of the mentioned editors can only be edited with the same tool and there is no way of using the development's native environments. Gade4all, on the other hand, offers the source code of the games, making possible for them to be modified directlty from the original development environment.
- Aside from having a graphical editor, Gade4all allows the access to the XML file that describes the complete request of the DSL.
- Games developed with Gade4all have a level-loading software module based on XML, which makes possible to add and edit levels without the editor.
- Gade4all doesn't require to type a single line of code, all the behaviors and rules of the game are already defined, which means that anyone can create games without problems, even people with no knowledge on software development. If the code of a level has to be modified, we can go directly to the generated project, but this will require knowledge on the development environment. However, other tools not only require previous programming knowledge, but it's also necessary to learn its own language. For instance, GML to develop with GameMaker.

E. Statistics of videogames developed with Gade4all

With the alpha version of this platform we have developed some videogames that have been deployed in different platforms like Google Play for Android and App Store for iOS applications. Table 1 shows the statistics of the games developed for the different platforms and in different languages. These are just some samples of the videogames that the "University of Oviedo" has published as an associate of the Gade4all project. The other associates of this project have also published their own games, with their own statistics.

TABLE 1: STATISTICS OF VIDEOGAMES DEVELOPED USING THE GADE4ALL TOOL

Product	Download	Platform
Tiro Skeet	2,113	Apple
Tiro Skeet	999	Android
Ace Athlete	1,475	Apple
Ace Athlete	6756	Android
Tennis Adventure	874	Android
Touch Tank Game	220	Android
Platforms Game	94	Android
Strategy game	68	Android
Trivia about Spain	8	Android
Pepins Adventures	9	Android
Puzzle Game	3	Android
Warzone	52	Android

V. CONCLUSIONS AND FUTURE WORK

With the use of the platform presented in this proposal we are trying to fill an important gap in the design and creation of videogames, seeking to simplify the development of this kind of applications for every user, even the ones without programming knowledge.

By following the approach of Model Driven Engineering, we have managed to implement a generic Domain Specific Language which has helped to abstract the main characteristics of the main videogame typologies. This DSL has served to automate all the process of creating videogames for multiple

platforms. This automation is reached through the graphical editor, allowing users to develop different types of 2D videogames in a simple and effortless way.

The Gade4all platform helps to increase the competitiveness levels in the development of videogames for mobile devices and the Web, as it makes easier to develop videogames for platforms like Windows Phone, Android, iOs and HTML5 without having to write a single line of code. With automatic generation of code, Gade4all improves the efficiency in the videogame development processes, reducing the development time and the number of errors and increasing productivity.

Currently, this tool can only generate videogames for the typologies shown in this proposal, but the development team is working to implement new ones, as well as improvements of some existing characteristcs, like the inclusion of a videogame library with some pre-designed videogames to make the development process even easier.

On its first stage of development, Gade4all only allows users to generate 2D videogames, but one of its objectives for the future is to support the creation of videogames in 3D.

VI. ACKNOWLEDGMENT

This work was performed by 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] M. D. Walker, M. Nilsson, T. Jebb, and R. Turnbull, "Mobile video-streaming," BT Technology Journal, vol. 21, no. 3, pp. 192–202, 2003.

[2] B. Selic, "MDA manifestations," The European Journal for the Informatics Professional. June, 2008.

[3] E. González, H. Fernández, and V. Díaz, "General purpose MDE tools," IJIMAI, vol. 1, 2008.

[4] S. Kent, "Model Driven Engineering," COMPUTER-IEEE COMPUTER SOCIETY-, vol. 2335, no. 2, pp. 286–298, 2002.

[5] T. Donovan and R. Garriott, Replay: The history of video games. Yellow Ant, 2010.

[6] M. Duggan, Making a GameSalad for Teens. Course Technology Ptr, 2013.

[7] J. Elliott, HTML5 Game Development with GameMaker. Packt Publishing, Limited, 2013.

[8] A. Baytak and S. M. Land, "A case study of educational game design by kids and for kids," Procedia - Social and Behavioral Sciences, vol. 2, no. 2, pp. 5242–5246, Jan. 2010.

[9] L. Stencyl, "Stencyl: Design Once, Play Anywhere," 2013. [Online]. Available: http://www.stencyl.com/.

[10] C. E. MONTENEGRO MARÍN, P. A. GARCÍA, J. M. CUEVA LOVELLE, and O. S. MARTÍNEZ, "Aplicación de ingeniería dirigida por modelos (MDA), para la construcción de una herramienta de modelado de dominio específico (DSM) y la creación de módulos en sistemas de gestión de aprendizaje (LMS) independientes de la plataforma.," DYNA, vol. 78, no. 169, pp. 45–52, 2011.

[11] A. van Deursen, "Domain-Specific Languages versus Object-Oriented Frameworks: A Financial Engineering Case Study," in Proceedings of Smalltalk and Java in Industry and Academia (STJA'97), 1997, pp. 35–39.

[12] E. Miotto and T. Vardanega, "On the integration of domain-specific and scientific bodies of knowledge in model driven engineering," Procs. of STANDRTS, vol. 9, 2009.

Edward Rolando Núñez Valdez Research fellow at Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering. Master in Software Engineering from the Pontifical University of Salamanca and B.S. in Computer Science from Autonomous University of Santo Domingo. His research interests include Object-Oriented technology, Web Engineering, recommendation systems, Modeling Software with DSL and MDA.

Oscar Sanjuán Martínez is a Lecturer at the Computer Science Department of the University of Carlos III of Madrid Spain. He is Ph.D. from the Pontifical University of Salamanca in Computer Engineering. His research interests include Object-Oriented technology, Web Engineering, Software Agents, Modeling Software with BPM, DSL and MDA.

B. Cristina Pelayo G-Bustelo is a Lecturer at the Computer Science Department of the University of Oviedo. She has a Ph.D. of the University of Oviedo in Computer Engineering. Her research interests include Object-Oriented Technology, Web Engineering, eGovernment, Modeling Software with BPM, DSL and MDA.

Juan Manuel Cueva Lovelle is a Ph. D. from Madrid Polytechnic University, Spain (1990). He is Mining Engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). From 1985 he is a Professor at the Languages and Computers Systems Area in Oviedo University (Spain). ACM and IEEE voting member. His research interests include Object-Oriented technology, Language Processors, Human-Computer Interface, Web Engineering, Modeling Software with BPM, DSL and MDA.

Guillermo Infante Hernández Have a Web Engineering Master degree 2010, from University of Oviedo. Has been working as research fellow in University of Holguin from 2006 to 2010 as part of the Computer aided Design and Computer aided Manufacture (CAD/CAM) research department team. From 2010 to the present he has been working on his doctoral thesis at the Computer Science Department of University of Oviedo. Some of the research interests he is working on include Model Driven Engineering applied to e-government domain, Rule driven software development and Web Engineering among others.