Núñez-Valdez, E.R., García-Díaz, V., Lovelle, J.M.C. et al. A model-driven approach to generate and deploy videogames on multiple platforms. J Ambient Intell Human Comput 8, 435–447 (2017).

DOI: https://doi.org/10.1007/s12652-016-0404-1

A model-driven approach to generate and deploy videogames on multiple platforms

Edward Rolando Núñez-Valdez[1] · Vicente García-Díaz[2] · Juan Manuel Cueva Lovelle[2] · Yago Sáez Achaerandio[3] · Rubén González-Crespo[1]

Abstract Currently, videogame development for mobile devices is a highly profitable and competitive industry worldwide. This profitability can be ascribed to the popularity that new technologies such as smartphones and tablets have acquired over the last few years. To increase competitiveness, we use a model driven engineering (MDE) approach to develop multi-platform videogames in a quick and easy manner. MDE helps us to reduce the number of errors, the amount of time required and the development costs and favors a productivity increment. The aim of this work is to present an approach for the agile development of multi-platform videogames by using high-level abstraction models. We offer a feasible solution supported by a tool that allows the development of videogames in a simple and agile way. An empirical validation is presented through a use case where the viability of our approach is shown in comparison with that of other tools. Finally, a survey shows the users' perception of the solution with the objective of validating the use case results. The given use case indicates that our approach is suitable for improving videogame development. The survey indicates that users think that it is possible to model videogames and generate the corresponding applications with our approach.

✉ Edward Rolando Núnez Valdez
edward.nunez@unir.net

[1] School of Engineering and Technology, Universidad Internacional de La Rioja (UNIR), La Rioja, Spain

[2] Computer Science Department, University of Oviedo, Oviedo, Spain

[3] Computer Science and Engineering Department, University Carlos III of Madrid, Leganés, Spain

Keywords Model-driven engineering · Software generation · User interface · Videogames

1 Introduction

The entertainment industry, particularly videogames, which also can be viewed as an activity that naturally engenders learning processes that contribute to the development of the player in many areas (Squire 2011), has become one of the main global markets (Marchand and Hennig-Thurau 2013). In addition, it promotes other complementary products such as processors, content, devices or broadband Internet access (Crandall and Sidak 2006).

Game production is a complex task due to the speed with which the industry evolves, the competitiveness of the market and the large number of existing platforms and programming languages to develop, deploy and distribute videogames. It is the combination of technology, game design and artistic content that leads to complexity and uncertainty in the development process (Ted Tschang 2005).

The complexity of software development inspired (Dijkstra 1972) to discuss the existence of certain problems related to how software was developed during the early years and propose new ideas about solving such issues and how software programming could be performed more appropriately. Nevertheless, currently, a large number of projects cannot be considered a success (Cerpa and Verner 2009), as they fail to meet the classic triple constraint of project management: time, budget and scope (PMI 2000). In the study presented by (Piraquive et al. 2015) shows an explanation why projects fail, analyzing different cases of projects and its variables. To avoid this constraint, the

software engineering community continuously offers new approaches for improving, as much as possible, the software development process life cycle, such as agile software development methodologies (Cockburn and Highsmith 2001) or continuous integration practices (Beck and Andres 2004).

One of the most promising changes related to software development over the last few years is the adoption, in both academic and industrial circles, of model driven engineering (MDE) as a real possibility (Kent 2002), focusing on raising the level of abstraction through the use of models instead of traditional programming languages such as Java or C++. Models allow for managing domain-related concepts of the problem and, through a series of automatic transformations, for generating software components understood by classic compilers and interpreters.

The aim of this paper is to present our approach to game development. After conducting a survey on the main types of videogames and their main characteristics, we extracted the common and different elements that can vary from one type of game to another, determining a schematic for the repetitive part of the domain under study that all model-driven developments have (Stahl and Voelter 2006). Using the main differences between videogames of the same genre, we have defined a metamodel that formally allows the creation of different models that conform to the metamodel, uniquely defining the characteristics of a type of game. This feature allows us to automate code generation for different platforms such as Android, iPhone, Windows Phone or HTML5 through the use of templates tailored for each.

The remainder of this paper is structured as follows: in Sect. 2, we present a brief overview of the relevant state of the art; in Sect. 3, we describe our approach for generating videogames under the principles of MDE; in Sect. 4, we discuss an evaluation of our approach using a case study; and finally, in Sect. 5, we present our conclusions and related future work.

2 Background

The evolution of mobile devices and the Web has ushered in a new era for videogame and entertainment applications. This ongoing evolution has promoted the existence of a variety of mobile devices and platforms that enable the development and execution of a great variety of applications. However, due to this wide range of platforms and devices, it is sometimes very difficult to develop applications for all of them in an efficient manner (Lavín-Mera et al. 2008). This is due to the differences that exist between the different mobile operating systems in the market, as each of them possesses specific APIs and use a different programming language.

As a result of this situation and with the objective of making videogame development easier and improving the development process, a number of graphical tools with support for the creation of videogames have surfaced on the market (Núñez-Valdez et al. 2013). Some tools such as *GameMaker*, *GameSalad* and *Unity* are intended for the development of applications for a variety of the abovementioned platforms.

These tools appear to make videogame development easier but actually present a high learning curve for inexperienced users as well as for developers using other programming languages. Moreover, they offer compiled solutions, with which the modification of and extension of the applications are impossible. It is also worth noting that these tools focus on the creation of the games' characters and levels but do not offer the user an easy interaction platform for creating other fundamental aspects of a videogame such as the menus, the score system or other common features; the definition of these other types of elements with the abovementioned tools remains a difficult task for the users.

In addition to these tools, some proposals have been made for improving the videogame development process. Furtado and Santos propose an approach for computer game development based on models and a created domain-specific language (Furtado 2006). With this approach, users are able to define aspects such as configuration, game states, game flow, exit conditions and basic properties. The generation was restricted initially to the C# language, focusing on videogames with audio components, sprites, items, characters, states and resources. The work is based on the implicit proprietary metamodel that is the core of the Microsoft DSL Tools Project (Cook et al. 2007a). Reyno and Carsí Cubel (2009) present an approach to 2D platform game prototyping automation through the use of MDE. This approach was used to prototype a Bubble Bobble type of game, including the structure and behavior of the game. It is worth mentioning that this work only presents one type of videogame modeling. There are other related works, such as (Solís-Martínez et al. 2015), where the approach is to model videogames using a reduced version of the business process modeling notation (BPMN) for modeling and generating videogames on different platforms. On another hand, García et al. (2014) proposes a parameterized transformation for a model of performance properties derived from a system model using MDE. The idea behind a parameterized term is to leave open the transformation framework to adopt future improvements and make the approach reusable. The reusable approach is used in the develop process of our platform.

3 Approach

In this work, we create a family of software products related to videogames, as they provide a prescription for the same core assets (Clements and Northrop 2002). In addition, we have divided this family into five smaller product families, one for each type of game we offer support for. Each of the variations in the family members are captured using feature models that describe hierarchies with mandatory features, options and alternatives (Kang et al. 1990). Thus, we can identify commonalities of and variances between the products of every family. MDE has been used on numerous occasions with families of products because models can be easily used to capture the differences that the products may have with respect to the other members of the family (Morin et al. 2009; García-Díaz et al. 2010). The solution is based on the idea of software product lines (SPL), which relies on the concept of software factories (Greenfield et al. 2004), with the aim of creating software through systematic reuse, supply chains and mass customization. There are several visions very similar to each other for creating a family of products based on models, related to which there is much scientific debate (Greenfield et al. 2004; Schmidt 2006; France and Rumpe 2007); however, they are beyond the scope of this work. All of them share the same principles and ideas. Thus, the four main concepts used in this work related to SFs are the following (Lenz and Wienands 2006):

- Architecture framework, which implements common features of a system and provides extension points where components can be integrated and extended.
- Product line development, which should only attempt to cover a specific domain or market segment, without attempting to cover all the possible domains.
- Model-driven development, which is closely related to the domain-specific language concepts. Based on a metamodel of the specific domain, users build models, and from them, templates are used to generate code for the different target platforms. In other words, the main principles of the MDE development approach are used in this work (Kelly and Tolvanen 2008a).
- Guidance in Context, which states that the SFs should include facilities such as code samples, how-to help pages, and so on.

The idea is not to create a system whereby we can create all types of applications automatically. Rather, SFs are focused on specific domains or families of software product lines. Conceptually, it can be divided into two main phases, the creation of a schema and the creation of a template of that schema. The SF schema is a model that can be interpreted by humans and tools, which describes work products, workflows used to produce the work products and assets used in the enactment of the workflows, for a specific family of software products in a given domain. Therefore, the development process depends on the experience of development because with a defective schema, the SF will fail. Thus, the SF template can be considered as an instance of the schema that is usually integrated into a development environment.

3.1 Game typologies

To develop games of different genres based on a model-driven approach, the first task performed was an analysis of the different types of existing games and their internal features. Users can develop any type of 2D videogame belonging to one of the following typologies: touch ability, puzzle, platform, trivia-style and turn-based strategy videogames.

Each of the typologies (smaller product families) is characterized by common elements (e.g., the existence of multiple levels, points recording, etc.) and similarities (e.g., settings, goals, objects, etc.). The common elements are easy to share among different videogames because they are always identical, except for small configuration aspects. However, similar elements have the same base but different behaviors, aspects or functions, and an analysis was required to discover what the differences are between one element and another. After we performed the analysis, we built a metamodel, which establishes a model that formally represents the possible variations among the elements of each videogame. Figure 1 shows a small excerpt of the inferred metamodel that shows the most representative elements for creating a game. Although the figure lacks certain details for reasons of space, the metaclasses indicate that a game can be of five different types (trivia, platform, puzzle, touch, strategy), and all of them have a specific configuration. The features will differ depending on the type of game chosen. For example, trivia games only focus on questions and elements, while platform games have different levels, characters, enemies and sprites. Obviously, games have different screens that will be similar between them but customized for every particular case. From the metamodel, final users are able to create models (using the tool we provide) that will be transformed to code for different platforms using specific templates.

Based on the metamodel, we have created an infrastructure that guides both the design and automatic videogame generation according to the guidelines established in the model-driven development principles (Stahl and Voelter 2006).

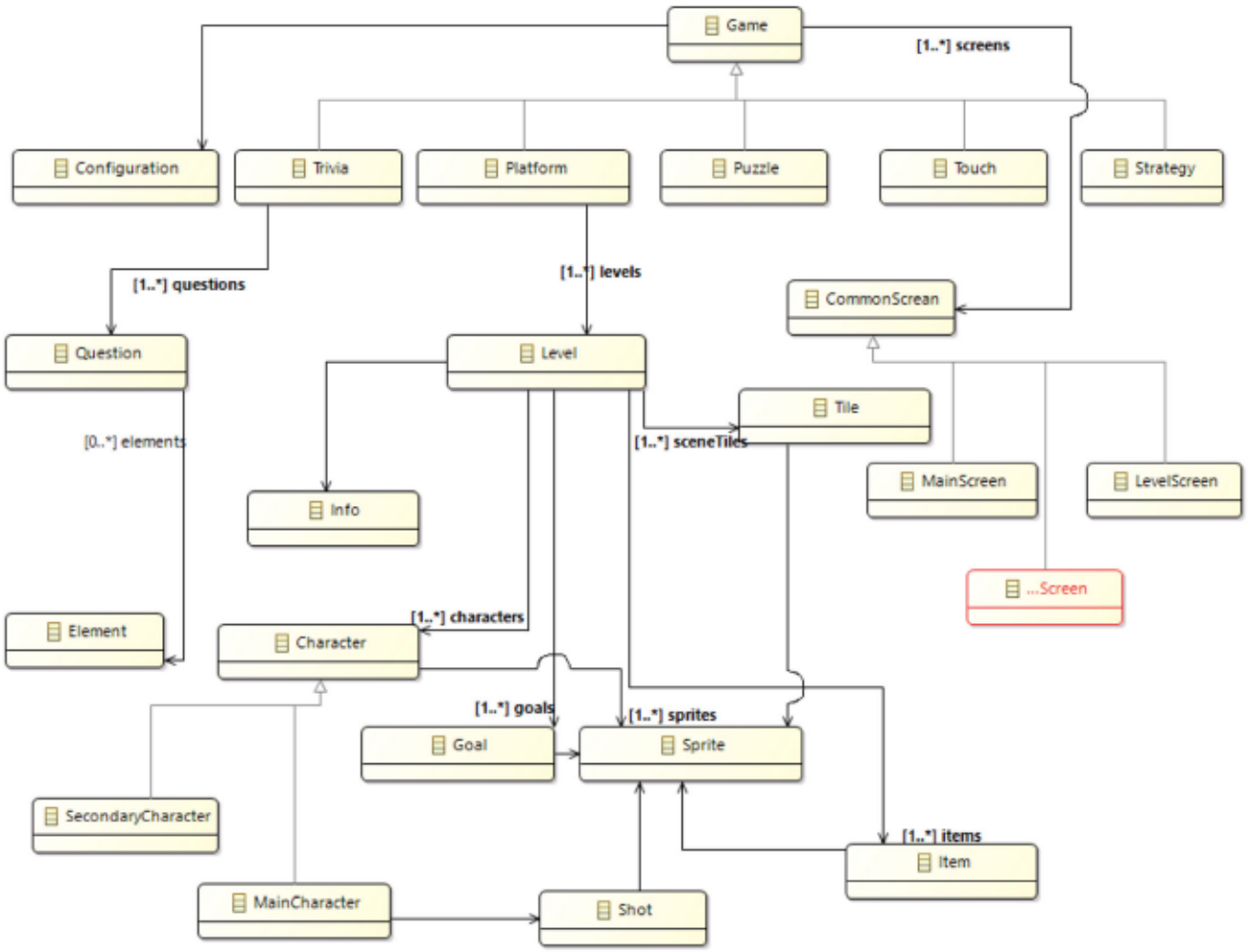

Fig. 1 Excerpt of the Gade4all metamodel for creating videogames

3.2 System architecture

To implement our approach, we built a system that relies on the common architecture of model-driven software developments, as shown in Fig. 2. The models created through the use of the editor are sent to the generation/transformation engine, which, after internal processing, outputs the source code for the videogames, making them available for execution on different platforms. The following sections detail the key components of the architecture.

3.2.1 Model

The graphical editor is aimed towards facilitating the task of the videogame designer and, at the same time, generating a view of the graphically designed element in XML format, representing the model of the system (e.g., size, damage, or design of a bullet using XML key-value pairs).

These files are generated internally through the use of the graphical editor so that the user is not required to be an expert. In MDE development, the model is the key element because it is the source from which all the final artifacts are generated using the transformation engine.

3.2.2 Graphical tool

All models and metamodels have an abstract syntax and one or more concrete syntaxes (Stahl and Voelter 2006). The most commonly used technologies related to MDE [e.g., those included in the Eclipse Modeling Project (Gronback 2009) or in the DSL Tools (Cook et al. 2007b)] use the XML language as their abstract syntax, with which it is easy to disseminate, interpret, persist or alter models through a large number of languages and tools. There are many alternatives for the concrete syntax, mostly textual or graphical, although there are other possibilities such as tree-based interfaces or combinations of these interfaces (Kelly and Tolvanen 2008b).

Our system also uses XML as its abstract syntax. In addition, to facilitate the work of users and avoid having to manually handle large and complex XML files, we have

Fig. 2 Architecture of the system

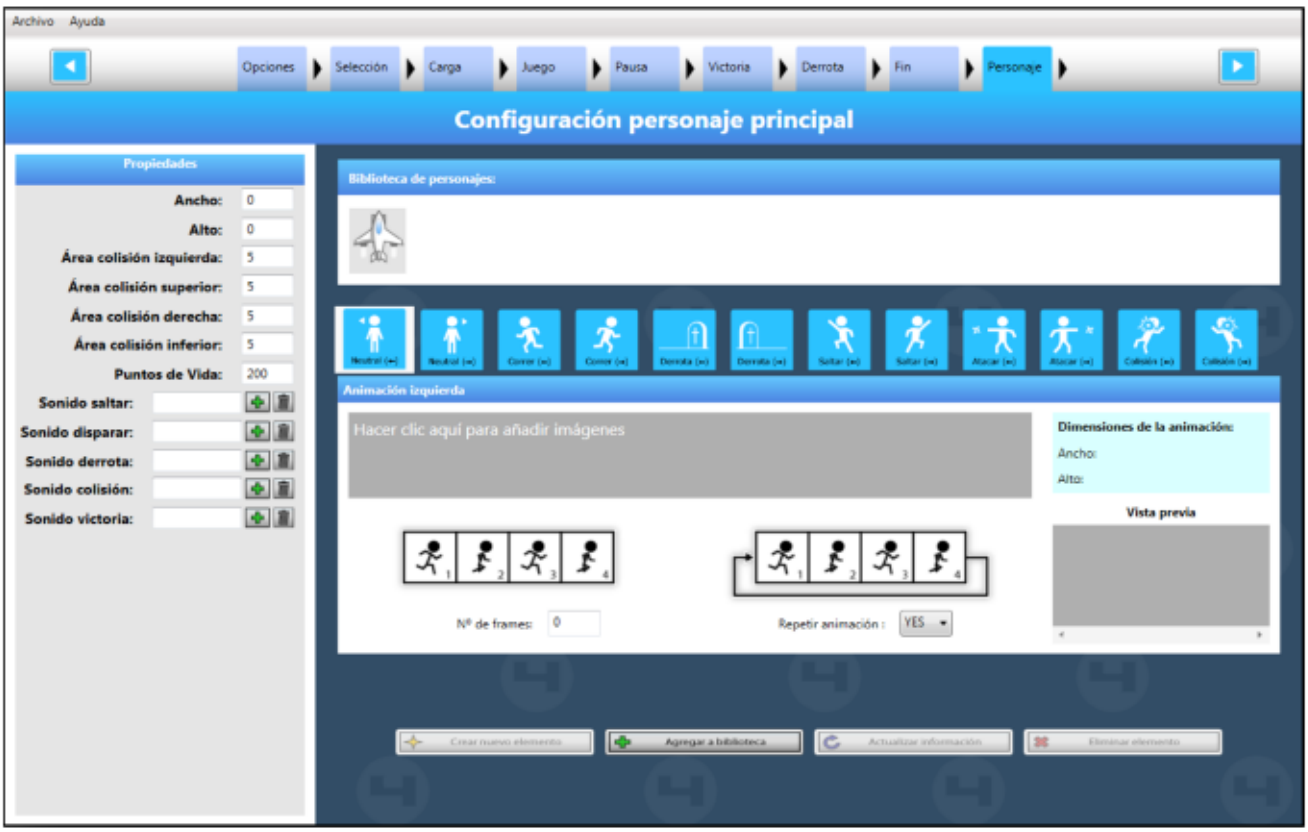

Fig. 3 Gade4all visual editor

chosen to provide a graphical tool inspired by other videogame generation tools previously cited in this paper. The Gade4all tool (Núñez-Valdez et al. 2013) has been developed following the MDE approach. The main objective of this tool is to facilitate agile videogame development. As show in Fig. 3, the visual editor is a layer between the users and the DSL to facilitate videogame development by people who do not have experience doing

```
public final static String imageButtonStart = %main_screen_start_button_image_source% ;
public final static int heightImageButtonStart = %main_screen_start_button_image_height% ;
public final static int widthImageButtonStart = %main_screen_start_button_image_width% ;
public final static int posXButtonStartMenu = %main_screen_start_button_x_position% ;
public final static int posYButtonStartMenu = %main_screen_start_button_y_position% ;

public final static String imageButtonExit = %main_screen_exit_button_image_source% ;
public final static int heightImageButtonExit = %main_screen_exit_button_image_height% ;
public final static int widthImageButtonExit = %main_screen_exit_button_image_width% ;
public final static int posXButtonExitMenu = %main_screen_exit_button_x_position% ;
public final static int posYButtonExitMenu = %main_screen_exit_button_y_position% ;

public final static String imageOptions = %main_screen_options_button_image_source% ;
public final static int heightOptions = %main_screen_options_button_image_height% ;
public final static int widthOptions = %main_screen_options_button_image_width% ;
public final static int posXOptions = %main_screen_options_button_x_position% ;
public final static int posYOptions = %main_screen_options_button_y_position% ;
```

Fig. 4 Excerpt of the template design for generating buttons

Fig. 5 Except of the template design for generating an enemy

```
<npcs>
 <Item>
   <npc_sprite_default_source></npc_sprite_default_source>
   <npc_sprite_left_source></npc_sprite_left_source>
   <npc_sprite_die_source></npc_sprite_die_source>
   <npc_width></npc_width>
   <npc_height></npc_height>
   <npc_x_position></npc_x_position>
   <npc_y_position></npc_y_position>
   <npc_sprite_attack_left_source></npc_sprite_attack_left_source>
   <npc_sprite_attack_left_width></npc_sprite_attack_left_width>
   <npc_sprite_attack_left_height></npc_sprite_attack_left_height>
   <npc_bullet_damage></npc_bullet_damage>
   <npc_bullet_has_intelligent_shot></npc_bullet_has_intelligent_shot>
 </Item>
</npcs>
```

so. However, the DSL can be used by developers with technical experience without using the visual editor.

3.2.3 Template

There are mainly two approaches for transforming the model into code (Czarnecki and Helsen 2003). On the one hand, the visitor-based approach focuses on using a visitor mechanism to traverse the internal representation of the model while writing the corresponding output. On the other, the template-based approach consists of a target text with some filling gaps to extract the information from the model and automatically populate it. For ease of use and for consistency with the technologies involved, in this project, we have adopted the most commonly used second approach. Figure 4 shows a fragment of the template needed to generate the buttons on the home screen of the videogame by filling empty spaces with the information extracted from the model. In addition, Fig. 5 also shows a fragment of the template used to generate an enemy. The former is created using a Java template, and the latter is created using XML. They are distinguished because some elements are going to be fixed and therefore must be embedded in Java code. In contrast, others are defined in XML files that are loaded at the beginning of the game, providing more flexibility to change the elements without recompiling the code, leading to the possibility of largely changing the game without the need for re-deployment.

The quality of the code generated for each videogame depends on the quality of the templates. Thus, the templates were created by expert developers who mastered the underlying technologies (Android, HTML5, Windows Phone and iPhone) through refactoring and refinements.

The quality therefore does not depend on the approach but rather on the experience of the designers of the templates.

3.2.4 Transformation engine

The transformation engine is a key factor in any MDE development (Sendall and Kozaczynski 2003; García-Díaz et al. 2015). It is used to convert models into other models or textual artifacts that could be, for instance, software that can be understood by any appropriate compiler or interpreter. In our case, the transformation engine receives a model as the input and, through a series of rules and transformations, creates code for the following platforms: Android, iPhone, Windows Phone and HTML5.

With any model that conforms to the Gade4all metamodel, the same transformation engine could generate artifacts for any current or future platform by simply using template specific to that platform. Reusability is one of the main advantages obtained when working with models (Mellor et al. 2003). To this purpose, we have several analyzers that are specialized according to the specific platform. One of their objectives is to read the DSL file and parse it, iterating through the whole model using an XML API.

4 Evaluation

With the objective of improving the validation of the results, the evaluation is done following the guidelines of an improvement case study proposed in (Runeson and Höst 2008). We have adopted the methodology so that it introduces a case study in a specific scope followed by a survey answered by the domain experts in order to understand their perception of the proposed approach. Our case study allows for establishing the viability of our videogame development approach, and the survey helps us determine if developers notice any benefit due to the adoption of our proposal during their development effort. The survey aims to validate the results of the case study and determine to what extent the participants agree with our approach.

4.1 Case study

The case study proposed in this paper aimed to investigate if the adopted approach is feasible in the videogame development process and if the effort level experienced by the user with the approach is acceptable. The following subsections offer a detailed description of the context, execution and results of the case study.

4.1.1 Nature of the case study

The objective of this case study is an approach to improve the development of multi-platform videogames; it be considered as an improvement case study (Runeson and Höst 2008). The subjects of the case study have some experience in videogame development and other types of applications on different platforms such as HTML, iOS, Android and Windows Phone. This means that we focused on offering the users a tool that would allow them to generate multi-platform applications quickly and without the need for programming, with the goal of increasing their productivity. The process followed during the case study allowed for the monitoring of certain variables, which allowed us to obtain some interesting results about our approach that will be discussed later in this paper. Moreover, due to the degree of software development experience of the users participating in the case study, we accounted for their perceptions of the development process when using the tool using a survey.

4.1.2 Research questions

The research questions included in the case study are the following:

- RQ1. Is the approach appropriate for improving the videogame development process? This question is related to the possibility of abstracting a conceptual model with the main features of the different videogame typologies. This abstraction is intended to serve as the basis for software reuse and automation of videogame generation. In this context, the concept of "appropriate" means that we want to raise the level of abstraction during development so that users do not need to work with low-level instructions that do not depend on the domain of knowledge (in this case, the videogame industry) but rather depend on the underlying technologies (concepts such as namespace, class, public, double, loops, and so on). Thus, users focus just on the main features of the different videogame typologies (main character, enemies, menus, tiles, targets, etc.).
- RQ2. Is the effort associated with the proposed approach acceptable? The answer to this question is based on measuring the effort users put into the testing procedure followed in the case study. With this metric, we intend to determine if the effort put into the development of a videogame with our approach is acceptable. In our case, the concept of "acceptable" means that users require less effort to create videogames than with other alternatives, taking into account that the videogames should have similar features and behavior.

4.1.3 Case selection

To select the case study, we evaluated the main tools in the market that have features similar to the one proposed in this study. This evaluation focused on selecting the most suitable tool for designing a videogame with both of the tools (our proposal and the commercial tool) in an efficient manner. The selected tool was GameMaker, as it is one of the main tools in the market and offers good usability rates for different user profiles.

Once the tools had been selected, the task to be completed with them was determined, accounting for the agility and efficiency of the task. Our approach clearly defines different videogame typologies, but the rest of the tools do not offer this possibility; thus, we decided that a platform videogame would be the best choice, as it is a classic typology and is available for both tools. Thus, in this part of the study, the user had to design and develop a simple platform videogame with three levels of different difficulty degrees (shown in Fig. 6).

4.1.4 Data collection procedure

The proceeding followed during the case study was divided into different steps or phases, to measure the proposed approach as precisely as possible. Once the metamodel had been defined and the tool had been developed, we decided to create a metric for the purposes of comparison of our tool with the other tools. The goal of this comparison was to determine if our approach achieved the objective initially presented.

During the testing procedure, support personnel were in charge of noting the number of mistakes or number of questions per user (when a user had doubts about the tools). The users' behavior during the task was automatically registered in order to measure the effort made by each user to complete the task. As is shown in Table 1, the monitored variables were the following: time elapsed, mouse clicks, keystrokes and distance traveled by the mouse.

The third step of the case study was to determine the most suitable user profile for carrying out the tests. The testing sample included 25 participants who had some experience in the development of videogames or other types of applications for different platforms but who had never used the tools included in the study. The users were offered a basic introduction to both tools and were given the resources (images, sounds, etc.) needed to complete the task. Furthermore, so as not to penalize or favor any of the tools, half the study participants started development using the Gade4All approach (13 out of 25), and the other half started development using the GameMaker tool (12 out of 25), preventing any prior development experience that could be decisive in the final results.

4.1.5 Result

With the objective of evaluating the videogame development process in the most accurate way with both tools (Gade4all and GameMaker), the following variables were monitored: time, errors, questions, keystrokes, clicks and distance.

Table 1 showcases the descriptive statistics of the results collected on the behavior of the users during the tests. In this table, we summarize the real values obtained during the test: we show a percentage comparison of the minimum, average, and maximum values per variable and tool for clarifying the differences between the tools.

Table 1 Descriptive statistics for the users' behavior

	Time (s)		Errors		Questions		Keystrokes		Clicks		Distance (inch)	
	G4	GM	G4	GM	G4	GM	G4	GM	G4	GM	G4	GM
Min												
#	1045	2936	0	1	0	1	72	631	684	1004	2897	4654
%	26.25	73.75	0.00	100.00	0.00	100.00	10.24	89.76	40.52	59.48	38.37	61.63
Average												
#	2089	3636	2	3	2	5	163	753	836	1134	3843	5823
%	36.48	63.52	34.17	65.83	31.76	68.24	17.80	82.20	42.44	57.56	39.76	60.24
Max												
#	3344	4969	4	6	5	9	289	910	989	1334	6318	6506
%	40.23	59.77	40.00	60.00	35.71	64.29	24.10	75.90	42.57	57.43	49.27	50.73

The main conclusions that can be extracted from the analysis of the results in Table 1 are the following:

- Taking as a reference the average time users needed to complete the task with each tool, we see that users needed, on average, 27.04 % more time with GameMaker than with the Gade4all tool. This indicates that videogame development time is reduced with our approach, resulting in an increase in productivity.
- If we compare the average number of mistakes made and questions asked by the users, our approach reduces the number of errors by 31.66 % and the number of questions by 36.48 %. This indicates that the Gade4all is more usable than GameMaker.
- From the clicks data, the keystrokes and the distance traveled by the mouse, we can see that the values associated with these variables are much smaller with our approach. Specifically, users needed 15.12 % less clicks and 64.40 % less keystrokes to complete the task, and the mouse traveled a distance that was on average 20.48 % shorter. With Gade4all, users needed to define a smaller number of elements because the main features of each typology are already included in this tool and the elements are better grouped, which reduces the distance traveled by the mouse.

4.1.6 Discussion

In this section, the results of the case study are discussed, with a focus on the research questions posed in Sect. 4.1.2.

1. RQ1. Is the approach appropriate for improving the videogame development process? As the obtained results show, the proposed approach is capable of improving videogame development, generally speaking. Through the use of MDE, we have managed to automatize the development of videogames for multiple platforms. This automation is possible with the use of the Gade4all tool, which allows any user to develop different types of 2D videogames in a simple and effortless manner. It can also be stated that with the automatic code generation, our approach increases the effectiveness of the videogame development process by reducing the time needed and the errors made and increasing the productivity.
2. RQ2. Is the effort associated with the proposed approach acceptable? One of the strong points of our approach is the fact that different videogame typologies were defined in the metamodel. This makes the abstraction of the main characteristics of each typology easier. The abstraction offers a set of predefined properties and features that are easy to modify during the development process. From the results of the tests, we could determine that the effort made with the Gade4all tool is less than the effort made with GameMaker, as the numbers associated with the monitored variables are notably lower (see Table 1). The reduction in effort is based on offering, by default, the definition of some important features of the videogame such as character behavior and design of the menu screens. This makes users focus on the development of the different levels of the game, which is the most important component of the process in terms of effort. These considerations and the results obtained allow us to determine that using our approach is acceptable.

As the results showcase in Sect. 4.1.5, the benefits of using MDE are numerous because they promote the development of more intuitive and better-designed tools that facilitate the creation of better applications in a reduced period of time. As the measured indicators show, the use of MDE improves the efficiency of the development process, reducing development time and the number of errors and increasing productivity.

4.2 Survey

After the case study, we proposed the validation of these results with a survey for the participants. The general goal

of this survey was to record the opinion of the users regarding the proposed videogame development approach. With the responses, we intended to determine if our approach could be adopted not only by professionals of the videogame industry but also by users without software development experience.

One of the most followed techniques when using surveys is the Likert type scale, which is used to determine to what degree the user agrees with a certain situation. In this survey, we adopted the five-point Likert scale with the following answer possibilities: 1 strongly disagree, 2 disagree, 3 somewhat agree, 4 agree, and 5 strongly agree. Table 2 shows the questions users had to, which were designed to determine their perception of the proposed videogame development approach.

4.2.1 Survey result

This section presents the results of the answers provided by the users in the survey. Table 3 showcases the descriptive statistics for better analysis and visualization of the results. Table 4 presents a summary of the results, offering a general view of the symmetry found in the distribution of the data.

After the case study and the survey, the main conclusions that can be drawn from the analysis of the results shown in Tables 3 and 4 and are the following:

Questions Q1, Q2, Q4, Q7 and Q12 yield the highest median values, which indicates these are the questions where users agree the most. As is shown in Table 4, one of the most important conclusions that can be drawn from the results is that 84 % of the participants concur completely on questions Q1 and Q4. It is also worth noting that 68 % of the participants strongly agree with questions Q2 and Q7. We also highlight that 52 % of the participants strongly agree and 32 % agree with question Q12. This last point is related to the native code generation feature of the tool, which enables developers to modify code written in a programming language they already know.

Table 2 Questionnaire

Profile	Question
Software Developer	Q1. Videogame modeling and creation of software solutions is possible with this approach
	Q2. This proposal helps to reduce the complexity of videogame software development
	Q3. Using this approach, inexperienced users can develop 2D videogames in a simple and intuitive manner
	Q4. This proposal makes multi platform videogame development easier, as it is only performed once and in one step
	Q5. This approach is suitable for videogame development but can also be applied to other domains
	Q6. Videogame developers can complete their work with fewer repetitions using this tool and, thus, reduce the number of errors and increase his/her productivity
	Q7. This tool helps users when generating applications for multiple platforms by reducing the development time, making the development process better
	Q8. This tool can reduce the costs of videogame development
	Q9. A videogame developer can easily adapt and extended the games generated with this tool because they are generated in native code for each of the platforms
	Q10. This tool can be considered as useful and usable, as it offers real support for videogame development
	Q11. It is easier and quicker to modify or add a new functionality to a videogame using the Gade4all tool than using a specific IDE for the given platform
	Q12. It is quicker to polish or improve an application generated with the tool than building it from scratch

Table 3 Descriptive statistics of the survey

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Min	4	4	3	4	3	2	4	3	2	3	2	3
Quartile 1	5	4	4	5	4	3	4	4	3	4	3	4
Median	5	5	4	5	4	4	5	4	4	4	4	5
Quartile 3	5	5	5	5	4	5	5	5	5	5	5	5
Max	5	5	5	5	5	5	5	5	5	5	5	5
Range	1	1	2	1	2	3	1	2	3	2	3	2
Inter qrt. range	0	1	1	0	0	2	1	1	2	1	2	1
Mode	5	5	4	5	4	5	5	5	5	4	4	5

Table 4 Response frequency to each question

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Total
Q1.						
#	0	0	0	4	21	25
%	0	0	0	16	84	100
Q2.						
#	0	0	0	8	17	25
%	0	0	0	32	68	100
Q3.						
#	0	0	5	13	7	25
%	0	0	20	52	28	100
Q4.						
#	0	0	0	4	21	25
%	0	0	0	16	84	100
Q5.						
#	0	0	6	13	6	25
%	0	0	24	52	24	100
Q6.						
#	0	1	6	6	12	25
%	0	4	24	24	48	100
Q7.						
#	0	0	0	8	17	25
%	0	0	0	32	68	100
Q8.						
#	0	0	2	11	12	25
%	0	0	8	44	48	100
Q9.						
#	0	3	5	6	11	25
%	0	12	20	24	44	100
Q10.						
#	0	0	2	14	9	25
%	0	0	8	56	36	100
Q11.						
#	0	2	8	8	7	25
%	0	8	32	32	28	100
Q12.						
#	0	0	4	8	13	25
%	0	0	16	32	52	100

Questions Q6, Q9 and Q11 are the most controversial questions on the survey, as they have an interquartile range of 2. This means that answers to these questions showed great differences, including some disagreements from the participants. For Q6, 4 % of the participants disagreed with the proposal, and for questions Q9 and Q12, the corresponding numbers were 12 and 8 %, respectively, which could be because some of the developers were not receptive to automatic code generation as error correction and functionality addition are more difficult when the application code is automatically generated. In addition to these disagreements, we must state that the 48 % of the users totally agreed with question Q6 and 44 % with question Q9. Additionally, 28 % of the users strongly agreed and 36 % agreed with Q11.

Questions Q3 and Q5 show neutral percentages and no disagreements. Only 24 % of the developers were uncertain about applying this approach to other domains (Q5), and 20 % were not sure about inexperienced users being able to develop 2D videogames in a simple and intuitive way with the tool (Q3). However, more than 75 % of the developers agreed or strongly agreed in both cases.

Questions Q8 and Q10 show that 8 % of the users were neutral participants and that there were no disagreements. This indicates that 92 % of the participants agreed or strongly agreed with Q8 and Q10.

5 Conclusions and future work

In this work, we have presented a platform for agile multiplatform videogame development by using high-level abstraction models. To this purpose, we have created a metamodel and a graphical DSL that allow people with no programming language knowledge experience to define different types of videogames. Using a generator, code for different platforms is automatically generated without user intervention.

The use of MDE standards and a graphical interface tailored to the needs of the end user suggest, based on the results obtained after conducting a quantitative study, that the approach is appropriate for improving the videogame development process because through the use of MDE, we have been able to automate the entire videogame development process for various platforms. The case study also shows that the effort associated with this approach is acceptable, as it helps generate less repetitive work in relatively minimal time. This means that our approach helps reduce the number of errors generated and increase productivity.

In addition, the survey answered by the 25 participants in the study reinforces the conclusions of the results obtained during the evaluation. Based on these results, users think that it is possible to model videogames and transform these models into software solutions using the proposed approach. Moreover, the participants stated that our approach makes polishing or improvement of the generated application quicker, which is directly related to the native nature of the code being generated. This feature enables developers to modify or extend their projects and create more complex applications in a shorter amount of time.

In future work, we plan to conduct more experiments, including the development of larger games and even videogames of other typologies, exploring also a possible integration with TALISMAN MDE Framework (García-Díaz et al. 2009). Additionally, we intend to include in the next case study users without videogame development experience, as the participants of the study noted that inexperienced users could benefit from the use of our tool; this point needs to be verified. Finally, the videogames are also used in learning context, e.g., the study presented by Baron et al. (2014) is an interesting work that shows an approach for continuous assessment of learning using videogames. In this context we plan conduct some experiments in learning area to improve continuous assessment process using the videogames generated with ours platform.

Acknowledgments This work was performed by the University of Oviedo under Contract No. MITC 11 TSI 090302 2011 11 of the research project Gade4all. We would also like to thank the reviewers for their insights and their comments that greatly improved the paper.

References

Baron HB, Salinas SC, Crespo RG (2014) An approach to assessment of video game based learning using structural equation model. In: 2014 9th Iberian conference on information systems and technologies (CISTI), pp 1 6

Beck K, Andres C (2004) Extreme programming explained: embrace change, 2nd edn. Addison Wesley Professional, Boston

Cerpa N, Verner JM (2009) Why did your project fail? Commun ACM 52:130 134

Clements P, Northrop L (2002) Software product lines. Addison Wesley, Boston

Cockburn A, Highsmith J (2001) Agile software development, the people factor. Computer (Long Beach Calif) 34:131 133

Cook S, Jones G, Kent S, Wills AC (2007a) Domain specific development with visual studio DSL tools. Addison Wesley, Boston

Cook S, Jones G, Kent S, Wills AC (2007b) Domain specific development with visual studio dsl tools. Pearson Education, Upper Saddle River

Crandall RW, Sidak G (2006) Video games: serious business for America's economy. Entertainment software association report, p 48

Czarnecki K, Helsen S (2003) Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA workshop on generative techniques in the context of the model driven architecture, pp 1 17

Dijkstra EW (1972) The humble programmer. Commun ACM 15:859 866

France R, Rumpe B (2007) Model driven development of complex software: a research roadmap. In: Future of software engineering (FOSE'07), pp 37 54

Furtado A, Santos A (2006) ALMS using domain specific modeling towards computer games development industrialization. In: The 6th OOPSLA workshop on domain specific modeling (DSM06)

García GM, Crespo RG, Martínez OS (2014) Parameterized trans formation schema for a non functional properties model in the context of MDE. Advances and applications in model driven engineering. IGI Global, Hershey

García Díaz V, Tolosa J, G Bustelo B et al (2009) TALISMAN MDE framework: an architecture for intelligent model driven engi neering. In: Omatu S, Rocha M, Bravo J et al (eds) Distributed computing artificial intelligence bioinformatics soft computing and ambient assisted living. Springer, Berlin, pp 299 306

García Díaz V, Fernández Fernández H, Palacios González E et al (2010) TALISMAN MDE: mixing MDE principles. J Syst Softw 83:1179 1191. doi:10.1016/j.jss.2010.01.010

García Díaz V, Pascual Espada J, Bustelo CPG, Lovelle JMC (2015) Towards a standard based domain specific platform to solve machine learning based problems. Int J Interact Multimed Artif Intell 3:6 12. doi:10.9781/ijimai.2015.351

Greenfield J, Short K, Cook S, Kent S (2004) Software factories: assembling applications with patterns, models, frameworks, and tools. Wiley, Oxford

Gronback RC (2009) Eclipse modeling project: a domain specific language (DSL) toolkit. Pearson Education, Upper Saddle River

Kang KC, Cohen SG, Hess JA et al (1990) Feature oriented domain analysis (FODA) feasibility study. Technical Report, CMU/SEI 90 TR 21,ESD 90 TR 222. Distribution 17, p 161

Kelly S, Tolvanen J P (2008a) Domain specific modeling: enabling full code generation. Wiley, Oxford

Kelly S, Tolvanen J P (2008b) Domain specific modeling: enabling full code generation. Wiley, Oxford

Kent S (2002) Model Driven Engineering. In: Butler M, Petre L, Sere K (eds) Integrated formal methods. Third international confer ence, IFM 2002 Turku, Finland, May 15 18, 2002 Proceedings. Lecture Notes in Computer Science, vol 2335. Springer, Berlin, Heidelberg, pp 286 298

Lavín Mera P, Moreno Ger P, Fernández Manjón B (2008) Devel opment of educational videogames in m learning contexts. Second IEEE Int Conf Digit Game Intell Toy Enhanc Learn 2008:44 51. doi:10.1109/DIGITEL.2008.21

Lenz G, Wienands C (2006) Practical software factories in.NET. Apress, New York

Marchand A, Hennig Thurau T (2013) Value creation in the video game industry: industry economics, consumer benefits, and research opportunities. J Interact Marketing 27(3):141 157

Mellor SJ, Technology P, Clark AN, London C (2003) Model driven development. IEEE Softw 20:14 18

Morin B, Barais O, Jézéquel J M et al (2009) Models@ run. time to support dynamic adaptation. Computer (Long Beach Calif) 42:44 51

Núnez Valdez ER, Sanjuan Martinez O, Bustelo CPG et al (2013) Gade4all: developing multi platform videogames based on domain specific languages and model driven engineering. Int J Interact Multimed Artif Intell 2:33 42

Piraquive FND, Crespo RG, Garcia VHM (2015) Failure cases in IT project management. IEEE Lat Am Trans 13:2366 2371. doi:10.1109/TLA.2015.7273799

PMI (2000) A guide to the project management body of knowledge (PMBOK guide). Project Management Institute, Newtown Square

Reyno EM, Carsí Cubel JÁ (2009) Automatic prototyping in model driven game development. Comput Entertain 7:1. doi:10.1145/1541895.1541909

Runeson P, Höst M (2008) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14:131 164. doi:10.1007/s10664 008 9102 8

Schmidt DC (2006) Guest editor's introduction: model driven engineering. Computer (Long Beach Calif) 39:25 31. doi:10.1109/MC.2006.58

Sendall S, Kozaczynski W (2003) Model transformation: the heart and soul of model driven software development. Softw IEEE 20:42 45

Solís Martínez J, Espada JP, García Menéndez N et al (2015) VGPM: using business process modeling for videogame modeling and code generation in multiple platforms. Comput Stand Interfaces 42:42 52. doi:10.1016/j.csi.2015.04.009

Squire K (2011) Video games and learning: teaching and participa tory culture in the digital age. Technology, education connec tions (the TEC Series). ERIC, New York

Stahl T, Voelter M (2006) Model driven software development. Wiley, Chichester

Ted Tschang F (2005) Videogames as interactive experimental products and their manner of development. Int J Innov Manage 9(1):103 131