A Mechanism of abstraction for independent definition of game's platform elements

Carlos Enrique Montenegro Marín[*]
Juan Manuel Cueva Lovelle[**]
Jordan Pascual Espada[***]

submitted date: September 2013
received date: October 2013
accepted date: February 2014

Abstract

This article is aimed at determining the feasibility of using a common mechanism to define the configuration in games for mobile devices. Such a configuration is intended to increase the level of abstraction up to a point where the definition of a platform game in XML format is made independent. This should then be read and deployed in an automatic way over any mobile platform.

Keywords

Abstraction, independent the definition, Mobil games, Domain Specific Language (DSL).

1. Introduction

One of the big problems that arise when generating applications for different mobile platforms is how to standardize the definition of common areas. For example, suppose that you want to display the same game for an Android device and for an IPhone device, the game has some common elements such as name, screen orientation, the main icon's game, the background image, additional images that will manage the game, sounds, button images, images of level change among others. Defining these elements in the appropriate

[*] B.Sc. In Systems Engineering, MSc. Information and Comunications, Universidad Distrital Francisco José de Caldas, Bogotá (Colombia). PhD. In Informatics Systems of Services for Internet, Universidad Pontificia de Salamanca (Spain).Current position: professor at Universidad Distrital Francisco José de Caldas, Bogotá (Colombia). E-mail: cemontenegrom@udistrital.edu.co

[**] B.Sc. In Mining, MSc. In Computers, PhD. In Computer Science. Current position: professor at Universidad de Oviedo, (Spain). E-mail: cueva@uniovi.es

[**] B.Sc. In Informatic Engineering, MSc. In Web engineering, PhD. In Computer Science. Current position: professor at Universidad de Oviedo, (Spain). E-mail: pascualjordan@uniovi.es

language for Android and for iPhone separately leads to carrying out repetitive tasks that could be minimized by reading information from a common format.

The approach presented herein attempts to solve such a problem. The idea is to read the common elements of an XML file in order to abstract these common elements in a single format that is accessible regardless of the platform. However, the problem is presented and is the purpose of this article to focus on studying the feasibility level of performance that this implies in different mobile platforms.

2. Gade4All Project description

The above problem is contained within the Gade4All project that seeks the goal of creating a Domain Specific Language (DSL), through which a person, without knowledge in developing applications for a specific mobile platform, can build his/her own games.

In this sense, it has been already established that the most relevant mobile operating systems today are Android and iOS, whose tasks involve the development of the game for each platform, considering the dynamism and the great insight that social networks have. A third platform that has also been considered, on which to deploy the game, is a web browser that uses HTML5 ((W3C)). Finally, and in order to provide greater coverage on mobile platforms, the game will also be deployed over Windows Mobile.

Therefore a working plan has been determined, which comprises the following stages:

1. Generic categorization of game mobile devices.
2. Establishment of the common elements among the different games by category.
3. Identification of common elements to all games.
4. The making of a game by category and different platforms, containing the common elements involved in each of the specific programming languages.
5. Performance of XML loader for each platform.
6. Establishment and deployment of a comparative metric to evaluate the operation of the game; reading the XML properties against the definition of these implicit properties in the same executable code of the game.

To perform the tests, an XML file that contains the features related to the levels of a mobile game will be created. This XML file will be loaded on each mobile platform (Android, iPhone OS and Windows Mobile) and will take some metrics regarding time to make a full comparative analysis. Therefore, it will be necessary to implement the same features on each mobile platform, obviating XML.

3 Mobile operating system and game creation software

This section is aimed at showing the reader a very simple description of the mobile platforms to be used and some frameworks that somehow resemble each other within the Gade4all project.

Android: One of the most famous mobile Operating System (OS) these days is Android. Introduced in 2008, this OS is a product of Google and has many features that a mobile OS should have. Initially, this OS was not so successful, having many bugs, e.g. Bluetooth file transfer was not supported. But developments continued and now Android is used by many mobile manufacturing companies.

The latest version of Android is Ice Cream Sandwich (version 4.0). This version combines the best of Honeycomb 3.0 for Android tablets with the Android Gingerbread 2.3 OS to create a single, united operating system that developers use for all Android devices. The tiny incremental changes we saw from Froyo to Gingerbread have disappeared completely. Ice Cream Sandwich heralds a dramatic growth. Despite its catchy name, Ice Cream Sandwich is all about strong lines, sharp corners, and darker colors, its applications are distributed through "Android Market" or "Google Play" [1].

iOS: Apple is a late entrant in the smart-phone industry, yet with a refreshing design and many innovations, Apple's iPhone quickly made inroads in the market. Accordingly, iPhone OS quickly became the number two smart-phone OS in the market. The iPhone is a closed smart-phone platform, running a closed OS; which means Apple has the full control on its hardware and software.

Apple plays the roles of device maker and vendor, OS owner and service platform maker. The iPhone is made by OEM companies and sold through Apple's stores. Apple gets a slice of profit from content providers and application developers through two platforms: iTunes and App Store. The iTunes sells music and movies to customers, and App Store sells application software [2].

Windows Mobile: The Windows Mobile operating system is available on multiple platforms, but the first devices to use Windows Mobile were Pocket PC's that were driven by the Pocket PC 2000 Operating System and powered by Windows CE 3.0, which was launched in April 2000. Pocket PC 2000 can support only screens with a resolution of 240 x 320. It was designed for early PDAs like the HP journey and the iPads. It was a CPU-specific OS so all the software created for these devices had to be targeted at specific platforms. After the Pocket PC 2000 came the enhanced operability and Smartphone compatibility of Pocket PC 2002, launched in October 2001.

Currently, Windows phone 7 is introducing a new interface for the design system that includes a codename which is known as "Metro". The screen itself has been provided with the links for applications, features, functions and individual items that include contacts, web pages, gaming applications or media items using tiles (i.e. click-able links). You can add, edit, rearrange and remove the links. These links also use the updater that will update the user interface using a real time system. Its applications are distributed through "Marketplace" of Windows Phone [1].

The <E-Game> Project: The <e-Game> project offers an authoring environment for educational adventure games that do not demand prior knowledge of Information and Communication Technologies. The author (an expert in a specific field) only needs to write documents that describe the contents of the videogame following the <e-Game> XML syntax and feed them into the engine. In turn, the engine produces a fully functional game from those documents [3]

In the Tutorial "Applying Domain-Specific Modeling to Game Development with the Microsoft DSL Tools", the authors introduce the concepts of domain-specific modeling (DSM) and domain-specific languages (DSLs), presenting how the theory can be productively put into practice with the Microsoft Visual Studio DSL Tools. The required steps for applying DSM to the game development domain are specified and illustrated with the creation of a visual DSL for modeling 2D adventure games. The ultimate goal is to allow game developers and designers to work more intuitively, with a higher level of abstraction and closer to their application domain [4]. Another contribution from these tools, in the same direction, is a report called "Using Domain-Specific Modeling towards Computer Games Development Industrialization" [5].

In the Project "Domain Specific Techniques for Creating Games", the authors examine strategic games and present the Game Design Language as a language for expressing game rules. The goal of this project is to apply domain specific programming techniques to create a language for generating turn based strategy games. The language is written in XML while the templates and runtime libraries for the architecture are written in Java. The programmer writes the players, boards, units, rules, and turn structure of the game in XML. The XML is given to a parser which fills in the templates to produce a game architecture. The programmer then needs to create a user interface for the architecture to have a game [6].

A report called "Developing Digital Games through Software Reuse" considers that a game title is software and thus faces the same restrictions of business applications. The authors intend to analyze, under the optics of reuse, if game development should resort to reuse, and where and how this happens. This survey analyzes digital game development aiming at identifying reuse techniques as well as the evolution of games since 2002. The report addresses questions like: did game engines become more productive and facilitate game development projects? Is the use of engines the only type of reuse in game engines? What is the similarity between game development and a software product line? What are the differences and similarities between game development and common commercial software? Is any formal reuse technique contemplated in the process of developing a game? [7].

The research work entitled "Sharpludus: improving game development experience through software factories and domain-specific languages" explores the integration of game development, an inherently creative discipline, with software factories, which are concerned with turning the current software development paradigm, based on craftsmanship, into a manufacturing process. A software factory specification (i.e. schema) for a given game development domain is described, and a domain-specific language (DSL) that completes part of the software factory specification is defined. Such concepts were implemented and deployed into a host development environment, which includes code generation (consuming a simple game engine) from diagrams specified by means of the visual DSL. Finally, real world scenarios, developed to validate the proposed software factory, and its visual DSL, show that the presented approach can be used by game developers and designers to work more productively, with a higher level of abstraction and closer to their application domain [8].

4. Test development

According to the outline of this work, the first step it is to categorize generic games for mobile devices. To this end, the classification made in the iTunes Store has been taken as benchmark (http://www.apple.com/es/itunes/). Also, we considered the group's experience in developing mobile games; thus we identified 5 types of games: Arcade / Action, Puzzle, Strategy, Trivia, Memory, Logic and Skill Touch.

In order to determine the common elements between the different games by category and the common elements of all the games, it was necessary to make an iterative process that consisted on developing a game by category and on each different platform. Each iteration of the process involved a comparative task that was carried out in order to determine the common elements. Table 1 shows the games performed and the specific platform on which each game was deployed:

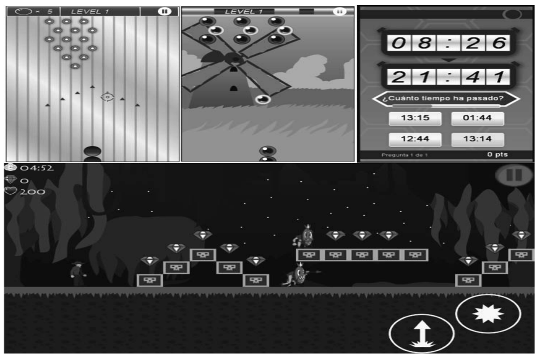
Table 1. Typology of games and platforms on which games were deployed

Typology	Platform
- Arcade / Action	Windows Phone 7
- Puzzle	iPhone
- Estrategy	iPhone
nm- Trivia, Memory, Logic...	HTML 5 + Javascript
- Skill Touch	Android

Source: own elaboration

Figure 1 shows the games created in each one of the platforms.

Figure 1. View of games in Iphone, Android, HTM5 and Windows Mobile



Source: own elaboration

Since each one of the platforms supported a different game, this exercise has allowed the abstraction of the configurable elements of the game in a class for each platform. Then, through a comparative process, debugging of the common elements of this class was achieved for all games on all platforms, these elements are shown below.

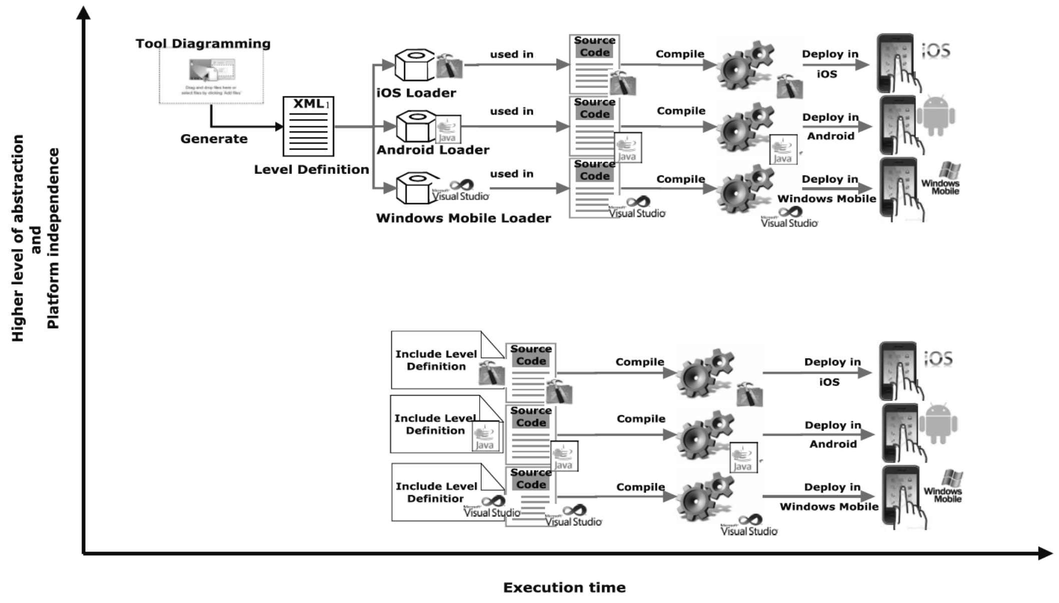
global_elemts: global_name, global_name_timespan, global_package_name, global_screen_orientation, global_icon, global_window_width, global_window_height, global_window_tile_width, global_window_tile_height, global_window_has_advertising.

screens_elemts: main_screen, main_screen_start_button, main_screen_options_button, main_screen_images, game_screen, game_screen_joypad_button, game_screen_shot_button, game_screen_jump_button, game_screen_pause_button, game_screen_status_bar, end_level_screen, end_level_screen_retry_button, end_level_screen_play_next_level_button, end_level_screen_images, select_level_screen, select_level_screen_next_level_button, select_level_screen_previous_level_button, select_level_screen_play_button, select_level_screen_level_thumbnail_view, select_level_screen_level_title_view, select_level_screen_level_images, end_game_screen, end_game_screen_return_button, end_game_screen_images.

Game: level, level_id, level_description, level_title, level_thumbnail_image_source, level_fps, level_attemps, level_end_level_condition, level_points, level_points_time_decrease.

The most significant contribution of this paper is summarized in Figure 2. This proposal is displayed as a tool that generates XML with the information necessary for playing, then this XML file is taken to a loader in each of the mobile platforms, then this loader is responsible for reading the information to be accessed on each platform. Then, each platform will measure the loading time, also performing the same process (i.e. obviating the XML). The idea is to determine the feasibility of extracting information that is common to all platforms in a file that is independent of and reusable for the platforms. This mechanism would increase the level of abstraction and gained independence, defining the levels of a specific platform.

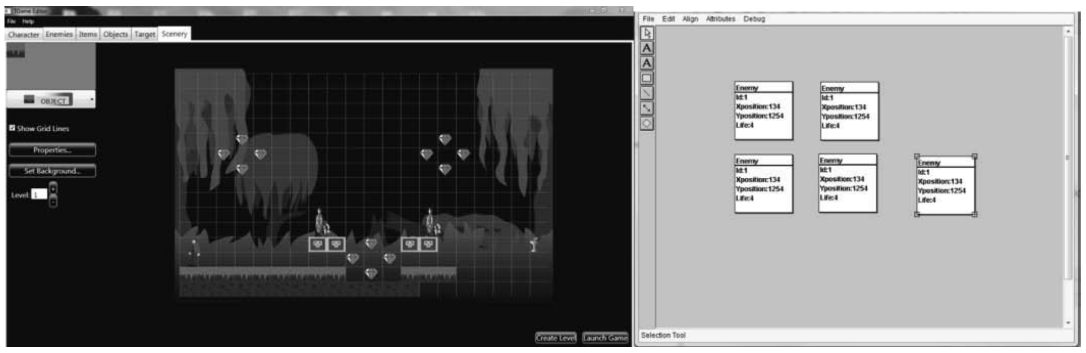
Figure 2. General vision of the process of abstraction of levels in a game for mobile devices regarding runtime



Source: own elaboration

The diagramming tool shown in Figure 3 consists of a palette that contains all the common elements defined in one level. To be integrated into the XML, users simply drag the elements onto the workspace and fill its properties. These tools have 2 versions, one in Visual Studio and one in eclipse.

Figure 3. Diagramming tool in Visual Studio and Eclipse



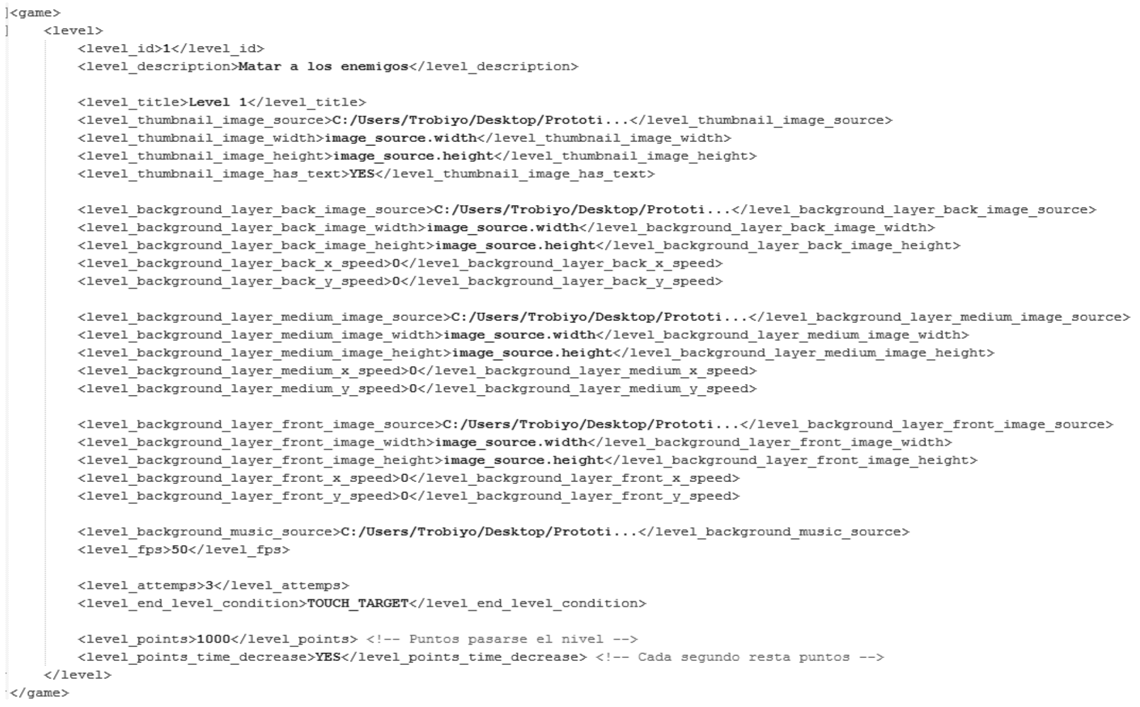
Enemy
Id:1
Xposition:134
Yposition:1254
Life:4

Source: own elaboration

The result of using the diagramming tool is an XML file that contains all the common and configurable information for games. Figure 4 shows the resulting file containing information on the level of a game and also of how the test of loading time was carried out. An important point here is that the example does not contain all the information on games, because this information is quite extensive and does not represent the main objective of this article.

Once the XML to be loaded is obtained, the next step is to implement the loader of this file for each platform. To this end, a mechanism has been created to load the XML file into iOS, Android and Windows Mobile. This loader will also be responsible for measuring the load times on each platform.

Figure 4. XML file that contains the level elements to play on mobile devices



```xml
<game>
    <level>
        <level_id>1</level_id>
        <level_description>Matar a los enemigos</level_description>

        <level_title>Level 1</level_title>
        <level_thumbnail_image_source>C:/Users/Trobiyo/Desktop/Prototi...</level_thumbnail_image_source>
        <level_thumbnail_image_width>image_source.width</level_thumbnail_image_width>
        <level_thumbnail_image_height>image_source.height</level_thumbnail_image_height>
        <level_thumbnail_image_has_text>YES</level_thumbnail_image_has_text>

        <level_background_layer_back_image_source>C:/Users/Trobiyo/Desktop/Prototi...</level_background_layer_back_image_source>
        <level_background_layer_back_image_width>image_source.width</level_background_layer_back_image_width>
        <level_background_layer_back_image_height>image_source.height</level_background_layer_back_image_height>
        <level_background_layer_back_x_speed>0</level_background_layer_back_x_speed>
        <level_background_layer_back_y_speed>0</level_background_layer_back_y_speed>

        <level_background_layer_medium_image_source>C:/Users/Trobiyo/Desktop/Prototi...</level_background_layer_medium_image_source>
        <level_background_layer_medium_image_width>image_source.width</level_background_layer_medium_image_width>
        <level_background_layer_medium_image_height>image_source.height</level_background_layer_medium_image_height>
        <level_background_layer_medium_x_speed>0</level_background_layer_medium_x_speed>
        <level_background_layer_medium_y_speed>0</level_background_layer_medium_y_speed>

        <level_background_layer_front_image_source>C:/Users/Trobiyo/Desktop/Prototi...</level_background_layer_front_image_source>
        <level_background_layer_front_image_width>image_source.width</level_background_layer_front_image_width>
        <level_background_layer_front_image_height>image_source.height</level_background_layer_front_image_height>
        <level_background_layer_front_x_speed>0</level_background_layer_front_x_speed>
        <level_background_layer_front_y_speed>0</level_background_layer_front_y_speed>

        <level_background_music_source>C:/Users/Trobiyo/Desktop/Prototi...</level_background_music_source>
        <level_fps>50</level_fps>

        <level_attemps>3</level_attemps>
        <level_end_level_condition>TOUCH_TARGET</level_end_level_condition>

        <level_points>1000</level_points> <!-- Puntos pasarse el nivel -->
        <level_points_time_decrease>YES</level_points_time_decrease> <!-- Cada segundo resta puntos -->
    </level>
</game>
```

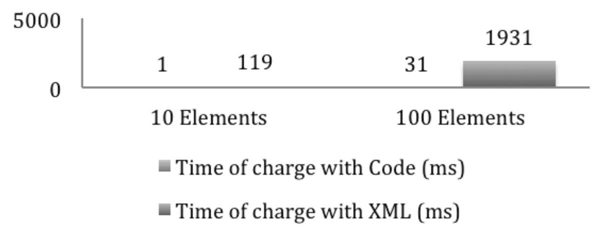
Source: own elaboration

Finally, the time it takes for the game to be deployed at the information level directly from the assembly is measured. These time measurements were used for comparison with the data-reading process from XML in order to assess whether carrying out the abstraction of the levels through an XML file that is independent of the platform is valid and does not involve a high cost in terms of run time. This issue will be discussed in the following section.

5. Comparative evaluation of level loading for a game in mobile devices through a platform-independent mechanism against a mechanism that loads the level in its own assembly

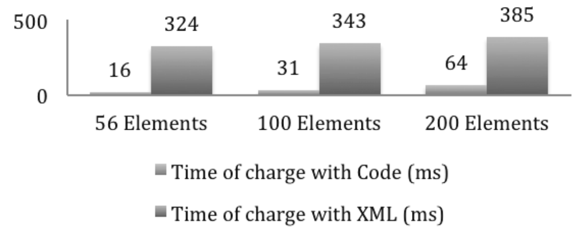
The level-loading times are larger than the time involved in reading information from the XML file, but the idea of this section is to determine if the time difference is not significant; thus the fact of using a mechanism that homogenizes and achieves platform independence in the definition of levels is more important. In this respect, we first show the times of each experiment, then make a comparison and finally argue if the proposal is valid or not, Figs. 5,6,7.

Figure 5. Generation time of a game when reading information from an XML file vs reading information from a source code in Android



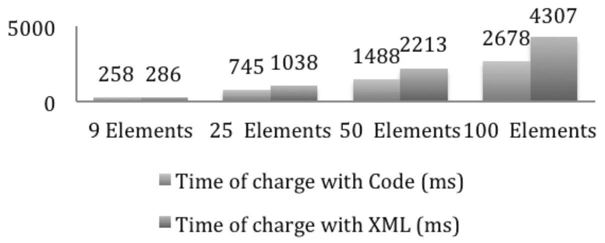
Source: own elaboration

Figure 6. Generation time of a game when reading information form an XML file vs reading information from a source code in Windows Mobile



Source: own elaboration

Figure 7. Generation time of a game when reading information from an XML file vs reading information from a source code in IOS



Source: own elaboration

Comparing the previous graphs, the best performance, when reading the configuration from XML, is observed for the Windows Mobile platform. This is because XNA has been used in the reading process of XML and this framework has provided enough improvement to read this type of files, whereas the other two platforms have made use of a manual charger, not involving any improvement.

The worst case is observed for the IOS platform when 100 enemies are loaded, showing an increment of 4307 milliseconds to load the game from XML. This means that the user must wait 4 seconds more for the game to be loaded in its mobile device; this time is acceptable as the charging time for loading will occur at the beginning of the game.

The longest wait time is observed in the IOS platform and the shortest time in the Windows Mobile platform. These differences are caused by the loading mechanisms and so reading files happens faster on one of the platforms. In this case the platform that provides better results is Windows Mobile due to the improvements included in the XNA Framework.

6 Conclusions

In conclusion, it is preferable to increase the level of abstraction of the elements that contains a level for a game, and make them independent of platform, since the price to be paid is only a small increase in the loading time of the game for the user. This in the sense that, with a single configuration file, games can be deployed on different platforms without redoing all the code and/or making it necessary to having expertise and prior knowledge about each specific platform.

References

[1] K. Vipin and G. Hitesh, “Mobile Operating Systems,” *IJEIR*, vol. 1, no. 2, pp. 115-120, April 2012.

[2] F. Lin and W. Ye, “Operating System Battle in the Ecosystem of Smartphone Industry,” *Information Engineering and Electronic Commerce, 2009. IEEC ’09. International Symposium on*. pp. 617-621, 2009.

[3] P. Moreno-Ger, I. Martínez-Ortiz, and B. Fernández-Manjón, “The< e-Game> project: Facilitating the development of educational adventure games,” Disponible en: http://www.e-ucm.es/drafts/11.pdf.

[4] A. L. de M. Santos and André W. B, Furtado, “Tutorial: Applying Domain-Specific Modeling to Game Development with the Microsoft DSL Tools”, 2006, disponible en: http://www.cin.ufpe.br/awbf/files/SBGames2006_TutorialGamesDSM. pdf.

[5] A. W. B. Furtado and A. L. M. Santos, “Using domain-specific modeling towards computer games development industrialization,” in *6th OOPSLA Workshop on DomainSpecific Modeling DSM’06*, 2006, p. 1.

[6] WORCESTER POLYTECHNIC INSTITUTE, “Domain Specific Techniques for Creating Games,” 2007.

[7] B. Neto, L. Fernandes, C. Werner, and J. Moreira de Souza, *“Developing Digital Games through Software Reuse,”* Journal of Information Processing Systems, vol. 6, no. 2, pp. 219-234, 2010.

[8] A.W. B. Furtado, “Sharpludus: improving game development experience through software factories and domain-specific languages,” Universidade Federal De Pernambuco (UFPE), 2006.