Developing Digital Games through Software Reuse

Beatriz Neto*, Lúcia Fernandes*,**, Cláudia Werner*
and Jano Moreira de Souza*

Abstract—Gaming is an old humans' habit. Games help in logical development and encourage learning of theoretical and practical concepts. Besides they offer entertainment and challenge. The advent of the personal computer changed this tradition. Every year new challenges arise in a digital format, which lead the young and adults to spend hours in front of a computer or TV screen in an attempt to overcome hurdles and reach an objective. Quality, sophistication, and constant innovation are attained through complex computer software that almost has an obligation to improve as each new title is released, due to this game development becomes a challenge. Considering that a game title is software and thus faces the same restrictions of business applications, this article intends to analyze, under the optics of reuse, if game development resorts to reuse, and where and how this happens.

Keywords—Games, Product Line, Software Reuse, Software Development

1. INTRODUCTION

One of the biggest issues in game development was the lack of game engine standardization [10]. Furtado and Santos [9] considered that the use of game engines provided a marked reduction in the lifecycle of game development, identifying reuse as the element responsible for the cost and time reduction, and the improvements in quality and productivity.

This survey analyzes digital game development aiming to identify reuse techniques and , the evolution of games since 2002: did game engines become more productive and facilitate game development projects? Is the use of engines the only manner of reuse in games 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?

These are some of the questions that influenced this survey, which is structured as follows: Section 2 deals with the evolution of games, Section 3 presents a brief review of formal software reuse concepts and Section 4 provides a comparison between game development and a product line.

The last section analyzes the results achieved and proposes new paths for study.

Manuscript received March 25, 2010; accepted April 16, 2010.
Corresponding Author: Beatriz Neto
* Systems and Computing Engineering Department, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil (beatrizneto@cos.ufrj.br, luciaaf@cos.ufrj.br, werner@cos.ufrj.br, jano@cos.ufrj.br)
** MI Montreal Informática Ltda, Rio de Janeiro, RJ, Brazil

2. From Sumerians to Marvel Heroes

The relationship between Man and games is quite old, and the oldest board game, identified as the "Royal Game of Ur", dates from the middle of the third millennium B.C. At the same time, the Chinese created a strategy game called GO, which was widespread within the eastern world. It was known as IGO in Japan and BADUK in Korea. Another example is the "Baoying-Xiangqi", considered as the oldest chess game [1].

Distinctly from nowadays, these games were created with specific objectives like strategic, concentration and prediction.

The speed of development and the spreading of games at that time was rather slow, whether because of the difficulties to manufacture the games parts or due to the communication difficulties that existed between the people.

Advances in this area became possible only with the Industrial Revolution, as large-scale manufacturing of the game elements became a reality. Many games were launched and quickly disappeared, such as Monopoly (1903) and Detective (1947), amongst others.

2.1 Modern Games

Board Games started to lose ground to videogames and other electronic entertainment forms in the 80s [CEILIKAN, 2009] and, with the creation of the first computer game (1952) and the first videogame (1958), the electronic game era began..

Gallery or Arcade games are units operated by chips, available in specialist shops. This game type was created in 1971, when Nolan Bushnell's Nutting Associates designed Computer Space.

In 1972 Atari built Pong, which was the first successful commercial videogame. This success has enhanced the game market. Titles such as "Football" (Atari, 1978), "Battlezone" (Atari, 1980), "Pac-Man" (Bally/Midway, 1980), "I, Robot" "(Atari, 1984), "STUN Runner" (Atari, 1989), "Virtua Racing" (Sega, 1992) and "Super GT" (Sega, 1997) were created at this time.

An evolutionary process of console games occurred in a similar manner, with the following examples: "Magnavox" (Odyssey, 1972); "Video Computer System" (Atari, 1977), "Game Boy" (Nintendo, 1989), "Super NES" (Nintendo, 1991); "Playstation" (Sony, 1995), "Nintendo 64" (Nintendo, 1996) "Playstation 2" (Sony, 2000), "Xbox" (Microsoft, 2001) and "Wii" (Nintendo, 2006) [24].

The year 2007 saw the reduction in numbers for exclusive digital games: many producers chose to go multi-platform since it was increasingly more expensive to produce a game and release it to just one system. That was the case with Koei who announced versions of "Fatal Inertia and Bladestorm" for the Xbox 360 which, until 2007 had been exclusive to PS3.

This seems to be one of the trends. Some classics were re-released in grand style in 2008, such as the remakes of Chrono Trigger for Nintendo DS, and Fallout 3 for PC, PS3, and Xbox 360 [UOL, 2009].

In May 2009, Raven Software's "X-Men Origins: Wolverine" game was released simultaneously for the PC, Xbox 360 and PlayStation 3 platforms [25].

3. SOFTWARE REUSE IN DIGITAL GAMES

3.1 Domain Analysis

Domain Analysis is a Reuse-based approach consisting of the identification and organization of knowledge on a class of problems – the problem domain – to provide support for its description and solution.

There are many difficulties in mapping and eliciting domain knowledge. An analyst could spend months working with experts in several different disciplines, only to become fluent in vocabulary of a given area. The following solutions may be applied to digital game development:

According to Cybulski and Reed [5], the RARE method (Reuse-Assisted Requirements Engineering) was created for requirement analysis and aimed at cataloguing, analyzing, and refining the requirements. RARE carries out its activities focused on 3 aspects. The first one is the identification and replacement of new requirements for already-modeled requirements. The second is the identification of requirements for new software that can be streamlined with re-usable components that were created in the development process of other software. The third is the identification of similarities between the requirements modeled in the same application domain that will point out the potential for reuse, conflict, or redundancy.

In order to meet these 3 aspects, the use of a requirement classification matrix is recommended to identify their similarities, allowing the creation of artifacts that support their refinement.

The domain lexical matrix classifies all the terms normally found in a particular problem domain. Table 1 describes 3 problem domains: the dice game domain, the coin game domain, and the general domain. The pre-classification of lexical terms assists the classification of the requirements that use it. The advantage in using such an approach lies in the fact that the lexical set of the problem domain is much smaller, if compared to the number of requirements that use each terminology.

Niu and Easterbrook [17] analyzed the functional and non-functional requirements of a software product line for a mobile phone football game. The goal was to efficiently identify and

Table 1. Domain Lexical Matrix. Adapted from [5]

Lexicon	Item Context			
Game Domain	Function	Method	Data	Environment
Card	--	value	--	instrument
Coin	--	value	--	instrument
Accord	start	--	interaction	Game
Dice	--	value	--	instrument
Coin-flipper	assign	value	--	Random
Shoot	end	--	--	Failure
Player	interact	value	--	User
Circle	arrange	value	random	--
Shake	arrange	value	random	--
Shuffle	arrange	value	random	--
Win	end	--	--	Success
Sign	assign	value	direct	--

Table 2. Conceptual analysis - football game product line

Module	#	Aspects	#	Interferences	#
Roles	3	Specific	2.7	utility	2
FRPs	17	Relevant/Pertinent	6	conflict	2
Drivers	7	Shared	6	variability	0
Scenarios	12	Irrelevant	2.3	coupling	5

develop the assets for a product line, and gain perception of the modularity of the requirements.

Formal conceptual analysis is a mathematical technique used to study the binary relations between sets. Applying this analysis to the software product line tackles two issues: identifying functional requirements of the product line (FRP) that contribute to system quality; and identifying possible interactions between functional and non-functional requirements.

The following elements were elicited in the football game analysis: roles of the user, developer and maintenance person; 7 drivers including both quality attributes typical of Software Engineering (SE), such as performance and maintainability, emotional attributes typical of videogames, such as excitement and frustration; and 12 scenarios that used 17 FRPs for the definition of the architecture drivers. Results are shown in Table 2.

3.2 Frameworks: Practical solutions

Frameworks can be used in the development of several applications in a specific domain, as described in the following:

Conejero and Hernández [4] proposed a framework to identify the cross features in the first stages of the development process and used it in the AGM product line. The working of a software product line requires the combination of a set of common and variable assets. Identifying these cross features allows the reduction of the dependency between them, resulting in a better reuse for a common-asset product family. The more independent and different the features are, less effort is needed to add new products to the product lines, as shown in Figure 1.

The analysis of the cross features consists of the following steps: execution of a feature-oriented analysis to grasp the main features of the product family, shown in Figure 2, identification of non-functional requirements; representation via a modeling language of the product requirements; creation of a dependency matrix where system features and the non-functional requirements are considered as the origin and system use cases are deemed as the goal.

Mattos, Rabello and Clua [13] created a framework for 3D first person shooter games (FPS), using the Microsoft XNA. The framework adds another abstraction layer to FPS game develop-

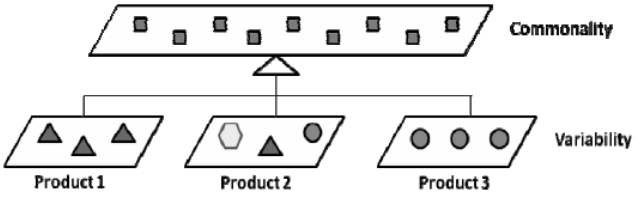


Fig. 1. Software product line characteristics

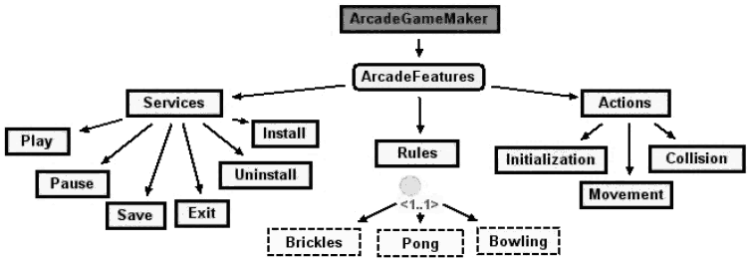


Fig. 2. AGM product family [4]

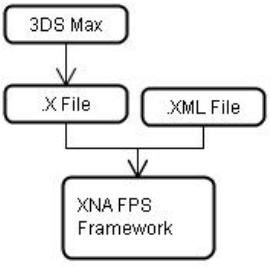


Fig. 3. 3D FPS game framework. Adapted from [13]

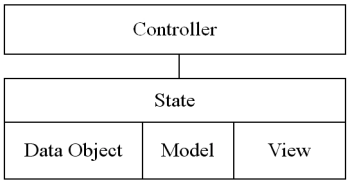


Fig. 4. 3D MVC pattern of FMMG framework [11]

ment, simplifying the process as it allows the developer to focus on the logics and dynamics of the game. Figure 3 shows the modules created and integrated to the framework.

The use of the framework consists of the creation of a project in the XNA Game Studio Express and the importing of the file that represents the scenario in a 3D model of the animated model files that represent the characters and of the XML file that refers to the source code to the attributes related to the object. The relationship between the elements is shown in Figure 3.

The Framework Mobile Multiplayer Games (FMMG) was created for the use in mobile multiplayer game architecture to allow developers to focus exclusively on game behavior. Its methodology relies on flexibility points – Hot Spot Driven Design. The hot spots are the parts of the framework that should be flexible to attend to the different domain applications [11].

FMMG uses three different project patterns: MVC pattern, state pattern and DataObject class. State pattern determines actual occurrence of a game based on player actions. DataObject class, which is responsible for data structure of state, will be manipulated by game (Fig. 4).

3.3 Digital Game Product Line Architectures

The importance of software architecture in systems development and in software reuse is that it produces solutions that increasingly involve the architectural level. Thus, optimization and innovation in the relationship between system components have become the main goals in the search for solutions that involve software reuse and digital game product lines, as discussed below:

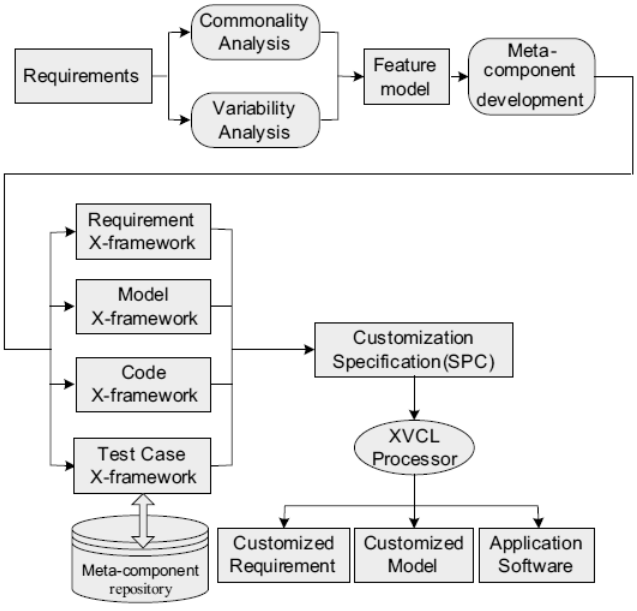


Fig. 5. Development Process in XVCL [22]

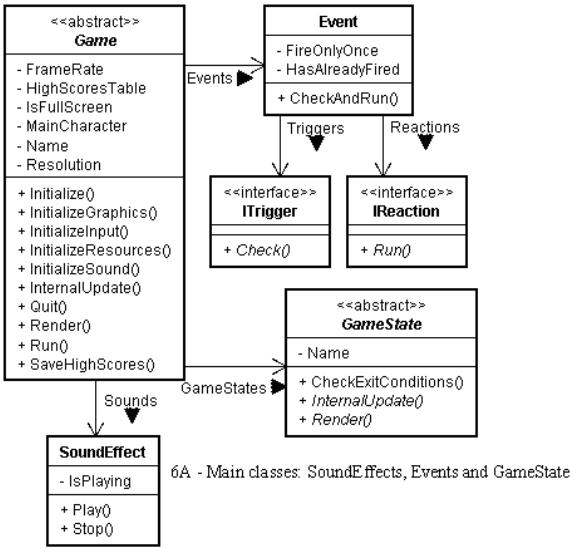
Zhang [22] employed the XVCL reuse techniques to obtain architecture with a uniform configuration for the construction and management of an RPG - RPG-PLA game line of products. XVCL is a programming meta-language, a tool and a method for the construction of reusable software assets (Fig. 5).

Product lines in the domain of mobile games have common features due to the properties of the domain and the restrictions of the business. Cho and Yang [3] classified these features as architecture standards: interface-oriented de-centralized control, centralized control, and adaptable contents.

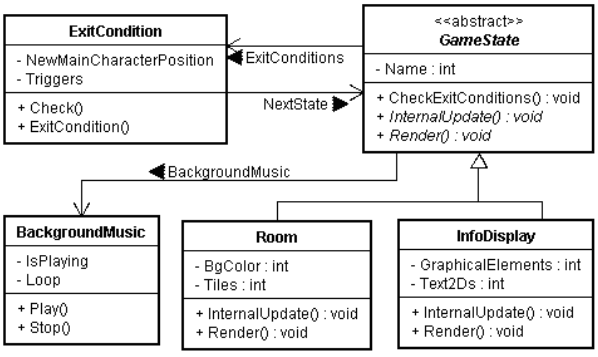
Furtado and Santos [9] presented SharpLudus, a software factory whose goal is a faster and more productive creation of 2D games through a product line. Figure 6 shows a game architecture produced by SharpLudus. In Figure 6a, main class game contains collections of SoundEffects, Events and GameState. Figure 6b details implementation of GameState with its ExitCondition and BackgroundMusic. Figure 6c shows an overview of basic components and animated character.

The most important extension brought by the SharpLudus factory to Visual Studio .NET is a new visual designer for the IDE allowing the developer to specify the configuration of the main class, the status and flow of the game, exiting conditions and properties. For that a visual representation was created of a Domain-Specific Language (DSL) and SharpLudus Game Modeling Language (SLGML).

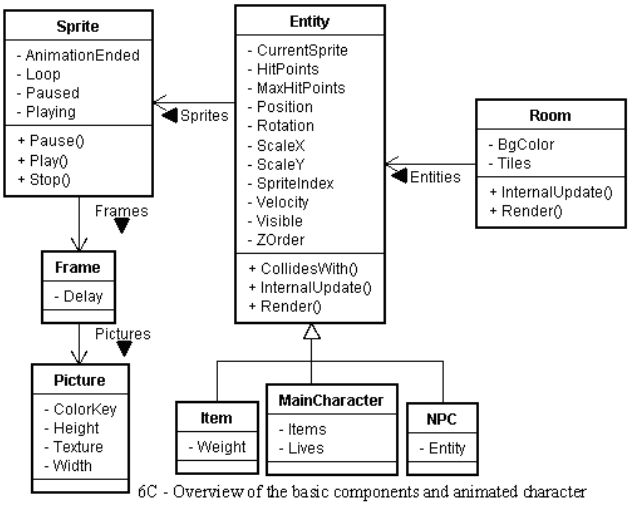
In the SharpLudus factory context, the game engine is a framework, the compiler is a code generator whose exit code feeds the game engine and the DSL programs can be changed into SLGML models.



6A - Main classes: SoundEffects, Events and GameState



6B - Implementation details of GameState



6C - Overview of the basic components and animated character

Fig. 6. SharpLudus game factory architecture. Adapted from [9]

3.4 Design Standards for Digital Game Product Lines

Design standards capture successful experiences making their reuse possible in similar problems. Some of the solutions proposed are presented below.

Nguyen and Wong [23] use design standards to grasp the rendering sub-system of a 2D board game. This sub-system processes the data structures of the 3D environment and executes the rendering from the gamer or the camera point-of-view [12].

Thus, the game domain, along with its rules, is separated from the type of visualization used, allowing the type of game to change independently from the way it is displayed to the user.

In the domain of engines, Vidal et al. [21] created a tool to grasp the graphical engine (3DState) of the virtual reality application. In this way, with the use of the Wrapper and Facade standards, it is possible to use the same engine in different types of realities and domains. Another approach in this area is the use of design standards for the abstraction of the Graphical API (DirectX) and scenario and mesh model construction.

3.5 Gaming Software Product Lines

Product lines represent a strategy to cut costs and improve the quality of software development processes. Some examples of these strategies are shown below:

Nascimento, Almeida and Meira [16] created a software product line based on 3 games: Monga, American Dad, and Zaak, as shown in Figure 7. For that, an approach for the creation of the product line was structured to meet the main aspects of the domain of mobile applications.

The implementation of this product line sets a model to identify its common and variable features, involving the definition and development of components using a subset or combination of several coding techniques. Its main stages are shown in figure 8.

Morin et al. [15] proposed to extend the Arcade Game Maker (AGM) product line with a new game. In the AGM product line, all the games are based on the principle that some elements are mobile whilst others are fixed, consisting of 3 games: Pong, Bowling and Brickles.

The extension of the AGM product line consists of the addition of a game strategy called “Command and Destroy” that is not so very different from the other games in the same product line, but requires adaptations in order to function. Through GeKo, an aspect-oriented generic composer, it is possible to change the behavior of mobile objects in AGM.

Vidal et al. [21] extended the SharpLudus software factory to create a product line and an ar-

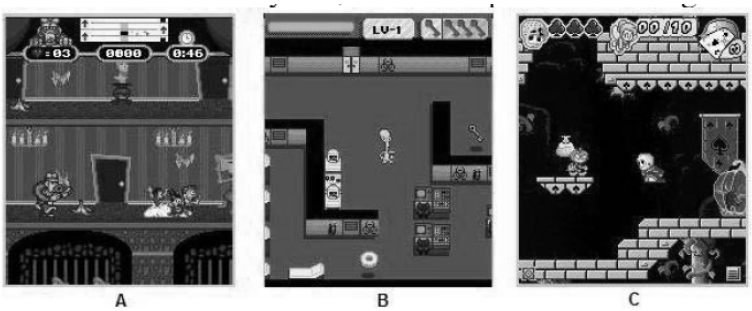


Fig. 7. Monga, American Dad and Zaak games

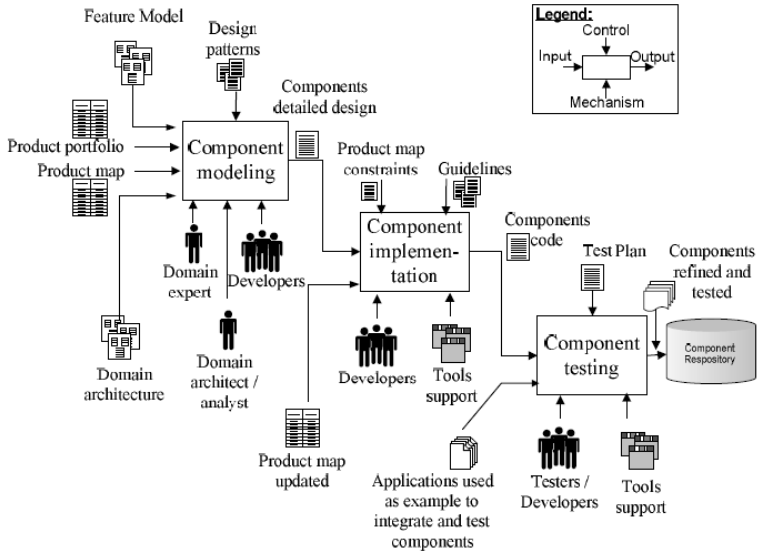


Fig. 8. Product line implementation stages [16]

chitecture for 2D mobile game development where the developer can use specific domain concepts to create visual models that can be transformed into source code [9]. In order to use this methodology, the code generator and the engine were reformulated to allow code generation for the J2ME platform.

Santos and Valente [19] implemented a software product line for mobile games using feature-oriented programming concepts (FOP) from the Bomber game. FOP is a modern technique for the modularization and separation of interests that allows the implementation of heterogeneous crosscutting requirements. The systems are built through the feature definition and composition that, in turn, differentiate systems of the same software product family. Figure 8 shows the features model for the Bomber game product family.

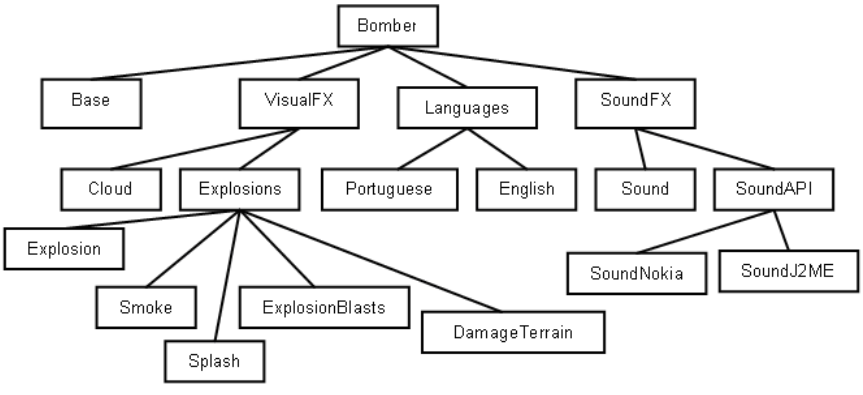


Fig. 9. Features model for the Bomber game product family. Adapted from [19]

For the creation of the product line, the features located in the leaf nodes of the model were identified in the source code of the game via manual comments for implementation modularization in the product line.

4. USING EXISTING ELEMENTS

This generation watches TV, listens to music, uses the mobile phone and notebook, all at the same time. It is highly familiarized with the Web and the PC. Nowadays it represents 50% of the active population (people aged up to 25) and, in 2020, it will be 80% of the population. This is a brief description of the Digital Natives, term coined by Marc Prensky to define those born after the 80s [14]. For this reason, they may be involuntary parties who are responsible for the use of the product line process in the game market, in order to attend the development of the same game for different platforms, especially mobile ones. So, table 3 below shows some game titles found for several platforms.

[1] - iPod nano (3rd and 4th generation), iPod classic, and iPod (5th generation).

One can see in Table 3 some specialized products of Apple Inc. The iPod line came from Apple's "digital hub" category [26], when the company began creating software for the growing market of personal digital devices. Video games are playable on various versions of iPods, but most of them were developed by Apple itself. With third parties like Namco, Square Enix, Elec-

Table 3. Different platform game examples

Title	PC	Playstation	Xbox	Wii	Cel	iPod[1]	iPod Touch / iPhone	iPad	Manufacturer
Age of Empires II	x	x	x		x	x	x		Microsoft
Age of Empires III	x		x		x	x	x		Microsoft
Battalion Wars 2				x					Nintendo
Blue Dragon	x		x			x	x		Microsoft
Brute Force	x		x						Microsoft
Captain Rainbow				x					Nintendo
FIFA 10	x	x	x	x	x		x	x	Electronic Arts
Final Fantasy	x	x	x	x	x		x	x	Square Enix
Halo 3	x		x						Microsoft
J.Bond: Quantum of Solace	x	x	x	x	x		x		Square Enix
Mario Kart				x	x	x	x		Nintendo
Need For Speed	x	x	x	x	x	x	x	x	Electronic Arts
Resident Evil	x		x	x	x	x	x		Capcom
Sonic	x	x	x	x	x	x	x	x	Sega
Street Fighter	x	x	x	x	x	x	x		Capcom
Super Mario	x	x	x	x	x		x		Square Enix
Tomb Raider	x	x	x	x	x		x		Crystal Dynamics
Wii Fit				x					Nintendo
Wii Sports				x					Nintendo
X-Men Origins: Wolverine	x	x	x				x	x	Raven Software

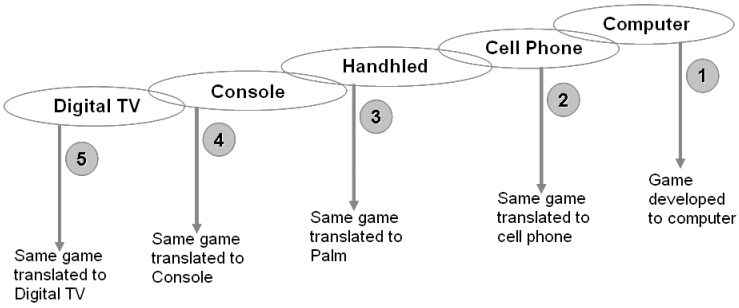


Fig. 10. Product Line Game possible Platform

tronic Arts, Sega, and others, all making games for iPod (iPhone), Apple's MP3 player has taken great steps towards entering the handheld video game console market [27]. We can see this even for a product (iPad), a new platform recently launched on January 27, 2010 [28], with games of third parties.

It is possible to see in Figure 10 that game migration between platforms is not only viable but also feasible.

Two analysis approaches were set from such observations: an external one where the observer does not know the SE processes and only compares the game products and construction stages; and an internal one where the observer considers the SE as the starting point.

4.1 External Vision

In this vision, for the analysis of reuse, the development stages and the many possibilities of game development found in industry were used (Table 4).

(1) A new product would be built from scratch; however, a strategy presently adopted is game derivation from films. Several titles refer to versions of blockbusters such as the Harry Potter and Spiderman series, amongst others. Reuse in this case is only associated with the idea, scenario, characters, and marketing.

(2) Products of the same series use existing characters, adding new and/or changing their resolution. The software can be extensively used if the game is of the same type as the original (i.e., fighting, adventure, and racing). This segment has chosen the construction of configurable games, generated from the same original, allowing the creation of a series

Table 4. Game presence on different platforms

Analysis Items X Reuse Methods	New Product 1	Series Products 2	Product Re-edition 3	Same Product on Different Platforms 4
Idea	little	none	much	much
Scenario	little	none	little	much
Characters	little	much	much	much
Software	none	little	little	little
Marketing	little	little	much	much

of sequels [6].

(3) The re-editing of a successful product can bring significant reuse advantages for the manufacturer as the idea and characters are pre-existent. However, pure and simple re-editing would not be attractive. In this way, the investment takes place in terms of higher quality and detail scenarios, and game platform updating.

(4). Nowadays it seems to be the path chosen by game manufacturers to gain better market penetration and ROI: the same game released on more than one platform. Developer companies can have full use of the idea, scenarios, characters, and the marketing, with the adaptations taking place only as regards to the change of platform. From 2006 Microsoft introduced greater development ease with the release of the XNA framework, allowing a game to be developed simultaneously and with the use of the same code for Windows and Xbox.

4.2 Internal Vision

Considering the SE vision, reuse and the games series, it was possible to see that the trend is the development of platform-based projects.

According to [18] the platform-based project is gaining space as it meets strong demands in the industry for efficient code reuse, modularization, risk reduction, and quick re-configuration. Games are assuming a platform structure, with a fixed component section and another with variable elements. Through the modification of the variables one can build a wide variety of customized games [7].

The changes to variable components are so common that they have even earned names: MODs, Add-ons, and expansion packs [7]. Table 5 below gives a picture of game software development:

(1) Virtual environments where game objects act. Uses specific technologies aimed at effects such as virtual sky generation (skybox) or water reflection [2]. Regarding the updates in the re-edition of products or the renewal of products of the same series, the reuse of scenarios loses strength. However, the use of the same scenarios in many platforms (i.e., Xbox, and Windows) is a guarantee of full reuse.

(2) Objects are models, usually 3D, that interact with the user and the scenario [8]. They are controlled by the engine or by user intervention. They form the characters, weapons, vehicles, etc. [7]. In the case of the same series products, it would be possible to change textures, models and even insert new characters. In product re-editing, this is one of the elements that undergo the most modifications to give the player the illusion of new software. Changing platforms, on the other hand, allows the use of all the original elements.

Table 5. Reuse in game software development

Item	Analysis Items X Reuse Methods	Series Products	Product Re-edition	Same Product on Different Platforms
1	Scenario	none	little	much
2	Objects	little	little	much
3	Interface (GUI)	none	little	much
4	Engine	much	much	little

(3) The graphical user interface (GUI) is the main element responsible for immersing the user in the game theme. As it is effectively linked to the engine, it may require some intervention in the programming language [7]. This element, despite its being more associated to the engine, keeps a strong relation with the game interface, following similar scenario reuse limitations.

(4) It is the nerve center of the software game, being responsible for the generation of 2D and 3D graphics (rendering), network applications, animations, collision detection, artificial intelligence, and audio [20]. The main engine can be understood as the encapsulation of several engines. A full engine can be developed as a platform, to attend to complete game lines. The Valve Corporation, manufacturer of the Half-Life game, recently developed the Source Engine for Half-Life 2 using Visual Studio and C/C++. This engine is capable of running under several environments (PC with MS Windows, Xbox 360 and Playstation 3) being used with hundreds of games [7]. The only restriction to this example is the use of some non-Windows platforms, such as Nintendo Wii.

Though most games for iPhone and iPod Touch should run on iPad, many developers are rapidly creating updated "HD" versions of their existing games to take advantage of the device's improved resolution and slightly altered layout. In this way, new product lines for HD games development will soon be created.

5. CONCLUSION

It has been pointed throughout this work that software reuse is constant in game development, especially regarding the engine that can, literally, be reused in the development of several games in the same platform without much (or no) effort.

It can be seen that game developer companies have an interest in launching their games for several platforms and this happens with sometimes more, or sometimes less effort. The Microsoft XNA framework offers as an advantage the generation of an engine that can be used in several platforms (PC, Xbox, Playstation and some mobile phone platforms). As for the migration of the product to the Nintendo Wii or another non-Windows platform, it requires an additional adaptation effort.

In games belonging to large console manufactures, such as Microsoft and Nintendo, the reuse is lower than in games belonging to smaller manufactures, as shown in Table 3. There is a commercial reason that is unrelated to reuse: there is major interest of selling the console and not just the game. Due to this, games became an attraction to sell consoles. Thus, game investments can be even greater since they can be compensated by the sale of new consoles. Since there are less console developers than game developers, we can assume that reuse is the main instrument used by the latter to make their product profitable.

We sought to assess the use of software reuse in this study as regards game development. We found that reuse does exist, although focused on the game domain.

Using the XNA as an example (to facilitate Windows-platform game development), it is possible to conclude that there are two product lines connected to the game: one focused on the game domain and the other focused on the framework (engine).

One product line geared for the framework could hold the stages (Fig. 2): "Requirements"

(where goal platforms would be defined),"Commonality Analysis" (where the common features of the chosen platforms would be analyzed) and "Variability Analysis" (where the differences and variations would be studied). The features needed to build a framework capable of meeting the needs of the desired platforms could be specified and/or customized based on such analysis. This framework would be the basis on which to build games.

This is a preliminary study that is to be refined in order to determine the features of a Framework-based Product Line.

REFERENCES

[1] P. Banaschak, "Early East Asian Chess Pieces: An overview," Aug., 1999.

[2] E. Bethke, 2003. "Game Development and Production". Wordware Publishing.

[3] H. Cho and J. Yang, "Architecture Patterns for Mobile Game Product Lines," In: International Conference On Advanced Communication Technology, 10, 2008, Phoenix Park. Proceedings. Washington: IEEE Computer Society, 2008. v. 1, pp.118-122.

[4] J. M. Conejero and J. Hernández, "Analysis of Crosscutting Features in Software Product Lines," In: International Conference On Software Engineering, 13, 2008, Leipzig. Proceedings. New York: ACM, 2008. pp.3-10.

[5] J. L. Cybulski and K. Reed, "Requirements Classification and Reuse: Crossing Domain Boundaries," In: Sixth International Conference on Software Reuse, 2000, Vienna, Austria.

[6] M. El-Nasr and B. Smith, 2006. "Learning through game modding, Computers in Entertainment," Nova Iorque: ACM Press.

[7] D. L. Figueirôa, F. Campos and A. M. Neves, "O Paradigma do Projeto Baseado em Plataformas Aplicado ao Game Design," In: VI Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital, 2007. (in portuguese)

[8] T. Fullerton, C. Swain and S. Hoffman, 2004. Game Design Workshop: Designing, Prototyping, and Playtesting Games. Lawrence: CMP Books.

[9] A. Furtado and A. Santos, 2006. "Applying Domain-Specific Modeling to Game Development with the Microsoft DSL Tools," In: 3rd Brazilian Symposium on Computer Games and Digital Entertainment (SBGames2006).

[10] G. Gelatti, "A Framework for Building Engines for Games and Simulations," I Workshop Brasileiro de Jogos e Entretenimento Digital, Sociedade Brasileira de Computação, 2002.

[11] M. M. Kubo, "FMMG: Um Framework para Jogos Multiplayer Móveis," 2006. 130 f. Tese Curso de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, São Paulo, 2006. (in portuguese)

[12] A. Lamothe, "Tricks of the Windows game programming gurus," 2. ed. [s.i.]: Sams Publishing, 2002. 1063 p.

[13] E. Mattos, B. Rabello and E. W. G. Clua, "Construindo um Framework para Jogos FPS com Microsoft XNA," In: Brazilian Symposia on Games and Digital Entertainment, 2007, Porto Alegre. Proceedings. Porto Alegre: Soc. Brasileira da Computação, 2007. (in portuguese)

[14] E. Monteiro, Nativos Digitais já estão dominando o mundo e transformando a forma como o ser humano se comunica. O Globo, 05/18/2009. (in portuguese)

[15] B. Morin et al. "A Generic Weaver for Supporting Product Lines," In: International Conference On Software Engineering, 13., 2008, Leipzig. Proceedings. New York: ACM, 2008. pp.11-18.

[16] L. M. Nascimento, E. S. Almeida, S.R.L. Meira, "A Case Study in Software Product Lines: The Case of the Mobile Game Domain," In: Euromicro Conference Software Engineering And Advanced Applications, 34, 2008, Parma. Proceedings. Washington: IEEE Computer Society, 2008. pp.43-50.

[17] N. Niu and S. Easterbrook, "Concept Analysis for Product Line Requirements," In: ACM International Conference On Aspect-Oriented Software Development, 8, 2009, Charlottesville. Proceedings. 2008: Acm, 2009. pp.137-148.

[18] R. Rucker, Software Engineering and Computer Games. Boston: Addison-Wesley Longman Publishing, 2002.
[19] R. C. Santos and M.T.O. Valente, "Extração de uma Linha de Produtos de Software na Área de Jogos para Celulares usando Programação Orientada por Features," In: Latin American Workshop On Aspect-Oriented Software Development, 2, 2008, Campinas, Proceedings. II Latin American Workshop on Aspect-Oriented Software Development, 2008. pp.50-59. Software Engineering Conference in Russia (SEC(R)2006). (in portuguese)
[20] R. Rouse III, 2001. Game Design Theory and Practice. Wordware Publishing.
[21] C. A. Vidal et al, "Ferramenta de Autoria de Ambientes Virtuais Adaptável a Diferentes Motores Gráficos," In: Simpósio De Realidade Virtual, 7, 2004, São Paulo. Proceedings. São Paulo: Simpósio de Realidade Virtual, 2004. pp. 15-26. (in portuguese)
[22] W. Zhang, "Architecturally Reconfigurable Development of Mobile Games," In: International Conference On Embedded Software And Systems, 2, 2005, Chengdu. Proceedings. Washington: IEEE Computer Society, 2005. v.//101109/200532, pp.66-72.
[23] D. Nguyen and S. B. Wong, "Design Pattern for Games," In: Technical Symposium In Computer Science Education, 33, 2002, Kentucky. Proceedings. New York: Acm, 2002. pp.126-130.
[24] A. L. Battaiola, 2000. "Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação" in JAI2000/SBC. (in portuguese)
[25] E. Stuart – X-Men Origins: Wolverine. Report published in the O Globo journal at 18 May, 2009. (in portuguese)
[26] Leander Kahney. Straight Dope on the iPod's Birth, Wired News, 2006-10-17.
[27] Mr. Marbles, Ahoy_And_Avast, & Vicious Sid - iPod Games Review Roundup - September 15, 2006
[28] Zumo Notícias - iPad: tablet da Apple é híbrido de iPhone com notebook – Report published at January 27, 2010 on http://tecnologia.terra.com.br/interna/0,OI4230761-EI4801,00.html (in portuguese)

Beatriz Neto

She is a Master of Science student in Systems and Computing Engineering at COPPE/Federal University of Rio de Janeiro (UFRJ). She has been a systems analyst at Serviço Federal de Processamento de Dados/Federal Service of Data Processing (SERPRO) since 2005. Her research interests are in the area of Innovation, Computer Supported Cooperative Work and Knowledge Management. They include topics such as Innovation Networks, Social Networks and Collaborative Systems.

Lúcia Fernandes

She is a Master of Science student in Systems and Computing Engineering from COPPE/Federal University of Rio de Janeiro (UFRJ). She has been a development manager at MI Montreal Informática Ltda since 2007. Her research interests are in the area of Database and Software Engineering, including topics like MDA (Model Driven Architecture), Education and Software Reuse.

Cláudia Werner

She received a Ph.D. degree in Systems and Computing Engineering from COPPE/Federal University of Rio de Janeiro (UFRJ) in 1992. She has been a professor at COPPE/UFRJ since 1994. Her research interests are in the area of Software Engineering, including topics such as Component-based development, Object-oriented Technology, Software Reuse and Domain Engineering.

Jano Moreira de Souza

He received a Ph.D. degree in Information Systems from University of East Anglia in 1986. He has been a professor at COPPE/UFRJ since 1976. His research interests are in the area of Database, including topics such as Knowledge Management, Knowledge Management of Scientific and Engineering, Decision Support Systems, Computer-Supported Cooperative Work, Autonomic computing, DBMS for Non-conventional applications, Geographic Information Systems.