A structured review of game coding through modeling

Diego Castro
PESC/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
diegocbcastro@cos.ufrj.br

Cláudia Maria Lima Werner
PESC/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
werner@cos.ufrj.br

***Abstract*—Game companies have grown a lot in recent years, reaching billions of dollars a year, despite the game-building process being very complex and time-consuming. This process has evolved considerably over the years. However, it has always been typically characterized by an ad hoc structure with little formal documentation. This lack of structure and documentation can cause severe problems throughout the game's development, such as delivery delays, wrong implementation, maintenance difficulty, among others. One possible solution that can accelerate this development process and increase the level of documentation is the use of modeling for automatic code generation. Based on this, the paper aims to perform a Structured Review to find some information on modeling for game generation, collaborating in game development processes by accelerating coding and assisting with documentation.**

***Index Terms*—Games, Development, Modeling, Code generator, Literature revision**

I. Introduction

There are millions of game developers globally, thousands of games created, and several companies that are working with games. Games have become one of the most popular entertainment sources, having fans of all genders and ages, becoming one of the highlights in the industrial environment and raising billions of dollars over the years [1] [2] [3].

Despite the growth of the game community in general, the process of development is typically characterized by an ad hoc development. In recent years, much effort has been made to introduce methodologies in game development. However, over the years, this process has become increasingly complex, requiring larger teams and requiring extensive documentation to maintain it. This documentation is not easy to maintain, which causes many communication failures and problems to be solved directly at the programming level, leading to complex maintenance of the games and low productivity [4].

Despite the efforts to improve the development process, creating a game can be very time-consuming, complicated, and involve several professionals, taking a long time to be built [5]. This long time of development is directly linked to the project documentation and the time that programmers take to code the games, create logic, phases, characters, and other objects.

Code generators aim to increase the quality of the code, decrease development time, generate code through standards, and maintain a high level of code consistency. Automatic code generation aims to provide a high-level code skeleton, leaving programmers to focus on specific aspects. Generators provide more productivity; delivering large volumes of code, which would take much longer if manually coded [6].

Code generators can speed up the game development process and can help with the documentation process. One type of documentation that is well known and used by programmers is modeling, which allows a general view of the system's structure. There are several types of modeling with different purposes: to visualize the structure, understand events over time, among others [7]. In software programming there are many tools, plugins, and Integrated Development Environments (IDEs) that support this development based on code generation through modeling, such as the easyUML, GenMyModel, Modelio and Eclipse Modeling Framework [6, 8].

This paper aims to present the main ways to use modeling in automatic generation of code for game development. The remainder of this paper is presented as follows: Section II describes the research method used in a structured review performed, Section III shows the results that were found, and Section IV concludes the paper with some final remarks.

II. Research Method

A structured review is a method that seeks to identify concepts based on structured stages of analysis to provide an overview on a given subject [9]. However, it does not follow all the quality criteria of a literature review. This method is mostly recommended for research where it is desired to get information about a topic in a faster way. The research process presented in this study covers papers that were published until April 2021. The search string was executed in Scopus as recommended by other studies [10]. The research questions, inclusion, and exclusion criteria used will be demonstrated below.

Research Questions

- **Q1:** What tools support game development through modeling?

- **Q2:** What are the advantages / disadvantages of using modeling to generate games?

Inclusion criteria

- The paper must be in the context of generating games through modeling;
- The paper must demonstrate practical concepts;
- The paper must provide data to answer at least one of the research questions;
- The paper must be written in English.

Exclusion Criteria

- Conference call;
- Studies that can not be fully accessed;
- Studies that are not in the area of Computer Science or Engineering.

The definition of the search keywords was made based on the PICOC (Population, Intervention, Comparison, Outcome and Context) strategy [18], using four of the five levels. The search string was defined by grouping keywords of the same domain with the logical operator "OR" and grouping the three domains with the logical operator "AND".

The search string returned a total of 307 papers. When analyzed according to the inclusion and exclusion filters, this number dropped to 5 papers. To minimize the lack of other search bases, considering that the study was only performed on Scopus, the snowballing procedure was used, which according to other studies [10, 19] can minimize article loss. The approach was applied, searching for new papers through the references (snowballing backwards) and through the papers that referenced these works (snowballing forwards). 140 more papers were analyzed and 6 more papers were included, totaling 11 analyzed papers.

Tables 1 and 2 show the search string used and the analysis process of the papers, along with the papers that were used for the review. Table 3 presents the papers that were analyzed. The Study Column demonstrates the information from where the paper was retrieved: M (Main study), B (Snowballing backward) and F (Snowballing forwards).

III. Results

A. Information found

From a brief analysis of the dates of the papers analyzed, it is possible to note that the topic is still a new topic, with recent works.

Game development is a field typically characterized by ad hoc development. However, much effort has been made to introduce development methodologies in this area of research [4].

From the model type column in Table 3, it is possible to observe that four ways of generating code through modeling were found, with Model-Driven Development (MDD) being the most used. Each of the types of modeling will be described in more detail below.

It is worth noting that not all the forms of modeling described below are modeled according to the development pattern known to most developers, such as UML. The modeling concept used in this work can be understood as any form of visual representation of the characteristics of an input.

B. What tools support game development through modeling?

The first form of modeling found represents a high level design where the developer can express his/her ideas through sketches that are transformed into code structures. In this type of modeling, there are mainly tools for 2D code generation [11]. Extending this idea a little bit, we found tools that allow real-time interaction with the game, where the user can interact with the narratives by drawing objects, which are then recognized by the system and converted into virtual objects in history, thus affecting the narrative's plot [12].

Another way found to generate codes through modeling is through Model-Driven Development (MDD). This approach is an emerging game development methodology that simplifies game development by reducing the gap between game design and implementation. For this type of modeling, different ways of use were found. However, they all revolve around two main steps: generating UML diagrams to describe the structure of the game through a specific language and generating game code structure from these models [13, 15, 16]. From this, the developer needs to define all the game's actions and, for each action, he/she describes the triggered events according to specific conditions. To model the game structure, a class diagram is used to demonstrate the players' characters, objects and the relationships between each one of them [4, 8, 8, 20]. The tools found is generic enough to be used in different types of games and to match 2D / 3D [13].

A product line is a set of systems with a group of similar functionalities developed on the same basis, making large-scale development more effective and productive. This strategy allows programmers to analyze and implement systems collectively through reusable domain assets, such as application blocks, structures, patterns, domain-specific languages, generators, and tools [17]. Following this method, a product line for arcade games was found. It used domain analysis processes to organize the game information and group the possible reusable components. From this, it was possible to create a base game with all the common features of arcade games and to specify new games with new features.

Finally, another form of modeling found is well-known by some game developers, being available within some engines. This type of modeling is called blueprint and can be understood as a visual scripting that provides an interface to build game functionality through simple-to-use behavior trees converted into code [14, 21].

C. What are the advantages / disadvantages of using modeling to generate games

There are different advantages and disadvantages depending on the type of modeling to be used to create the game. The first advantage of using modeling for automatic code generation

TABLE I
SEARCH STRING

P	Game*
I	UML, *diagram*, sketch, modeling
C	Not applicable
O	tools, approach*, method*, ideas, framework*, interpretation*
C	create, creation, production, development, elaboration, generation, practice
TITLE-ABS-KEY ((*game*) AND (UML OR *diagram* OR sketch OR modeling OR blueprint) AND (tools OR approach* OR method* OR ideas OR framework* OR interpretation*) AND (create OR creation OR production OR development OR elaboration OR generation OR practice)) AND (LIMIT-TO (SUBJAREA ,"COMP") OR LIMIT-TO (SUBJAREA , "ENGI"))	

TABLE II
ANALYSIS OF THE PAPERS

Activity	**Main Study**	**Number of papers**	**Snowballing Backwards**	**Number of papers**	**Snowballing Forwards**	**Number of papers**
Repeated Papers	307 added	307	94 added	94	46 added	46
Papers in another language	4 withdraw	303	2 withdraw	92	2 withdraw	44
Remove books	11 withdraw	292	6 withdraw	86	0 withdraw	44
Remove by title	241 withdraw	51	57 withdraw	29	22 withdraw	22
Remove by abstract	42 withdraw	9	21 withdraw	8	16 withdraw	6
Papers not found	0 withdraw	9	0 withdraw	8	0 withdraw	6
Remove by full paper	4 withdraw	5	4 withdraw	4	4	2
Total papers included	**5 papers**		**4 papers**		**2 papers**	
11 papers						

TABLE III
TRACEABILITY MATRIX

Title	**Authors**	**Year**	**Q1**	**Q2**	**Modeling Type**	**Study**
Generating Stable Building Block Structures from Sketches	Stephenson et al.	2021	X	X	Sketches	M
Sketch-based interaction for planning-based interactive storytelling	de Lima et al.	2020	X		Sketches	M
A framework for the development of serious games for assessment	Hamiye et al.	2019	X	X	Model-Driven Development	M
Modeling and Code Generation of Serious Games for Assessment	Hamiye et al.	2018	X	X	Model-Driven Development	B
Reinforcement learning for all: An implementation using unreal engine blueprint	Boyd and Barbosa	2017	X	X	Blueprint	B
A model-driven framework to support development of serious games for game-based learning	Tang and Hanneghan	2015	X		Model-Driven Development	B
A dsl for rapid prototyping of cross-platform tower defense games	Sánchez et al.	2015	X	X	Model-Driven Development	F
A Modeldriven serious game development integration of the gamification modeling language gaml with unity	Matallaoui et al.	2015	X	X	Model-Driven Development	F
Improving digital game development with software product lines	Furtado et al.	2011	X	X	SPL	F
Accelerating the creation of customized, language-specific ides in eclipse	Charles et al.	2009	X	X	Model-Driven Development	B
Model-driven game Development: 2D platform game prototyping	Reyno and Cubel	2008	X	X	Model-Driven Development	M

is its fast and intuitive way of using it; without specialized knowledge or programming skills, it would be enough for the professional to understand modeling [11]. Generators can still provide: more productivity; rapid prototyping; large volumes of code, which would take much longer if manually coded, among others [6].

As already mentioned, the process of developing a game is mostly done in an ad hoc manner, requiring many problems to be solved directly into the code. This makes documentation difficult, leaving it out of date and not reflecting the reality of the game. The advantage of using modeling for programming is that the model will always be updated, making it easy to understand the project.

Mainly focused on using product lines that are still possible: using repositories on a segment of the market, potential reduction in the number of errors with a large part of the code being reused, incremental development, etc. Table 4 presents a summary of the advantages of each type of modeling presented.

Despite the advantages offered by the code generators, most of the modeling/development tools presented are used as something complementary for coding (except for the blueprint). These tools only generate a general structure of the code to be created, saving initial development time for developers [4, 11, 12, 13, 14].

D. Discussion

The study sought to find information on how software modeling could facilitate and help game implementation. Throughout the previous section, some strategies that could help in this scenario were demonstrated. The main purpose of these strategies would be to use them in some visual way for automatic code generation.

TABLE IV
ADVANTAGES OF EACH MODELING METHOD

	MDD	**Sketches**	**SPL**	**Blueprint**
More productivity	X	X	X	X
Rapid prototyping	X	X		X
Generate large volumes of code	X	X		X
Incremental development	X		X	X
Less bugs	X		X	
Updated documentation	X	X	X	X
Game modification	X		X	
It is worth noting that the level of documentation of the tools are different and according to the modeling.				

Nowadays, hardly a person will create a game without the help of an engine. These engines already have different ways of modeling that facilitate coding and automatic code generation, sometimes even being specific to a topic, such as graphics, animations and script generations. However, these ways of modeling can still be improved.

Regarding code generation for a single project, the blueprint tool can help a lot in coding, being possible to create entire games with it alone. However, some papers report some limitations of this tool regarding what one can do with it, often not being possible to do very advanced things, requiring explicit programming. The code generation tools through Sketches follow the same pattern on what it is possible to create by using them, but they are also limited about the need for more advanced functions to create games.

There are two ways to create software: to have an idea and develop it from scratch or to build on something existing to create a new one. Many games are created based on existing games. Blueprints and Sketches tools aim to create code through their modeling. However, they are not directly concerned with the scalability of the game's expansion or the creation of games from an existing one. From Table IV, it is possible to observe that the two tools that mostly support the idea of creating a game and adapting it to create new ones are the MDD and SPL tools.

Current engines do not have systems that support this idea of game expansion and rapid modification. This is also an important way of creating software, where a program is adapted to generate a new one [22]. Currently, it is necessary to expand the game in a way that new stages, characters, and mechanics are modified directly in its source code. By using support tools with MDD and SPL approaches, it would be possible to create games where their expansion would happen faster and that are more reusable. So that the main core of the game would be created and it could be expanded in several different ways, always preserving the main features of the game.

Depending on how one wants to create a game, different approaches can be used. Some that already exist in game engines themselves and others that still need to be thought about and supported by these engines.

IV. Final Remarks

As previously mentioned, the process of building games is very complex and mostly happens in an ad hoc manner, causing many problems to be solved directly in code, leaving the documentation inconsistent with reality. In this context, it is possible to point out two general problems: a long time to build a game and lack of documentation.

This paper presented the information found in a review on the use of code generators through game modeling. Through this information, it was possible to observe four types of modeling, each with its particularities, advantages, and disadvantages, with emphasis on the MDD method that was the most used. Among the main advantages, the following stands out the gain in speed in the development and continuously updated documentation, thus solving the two general problems demonstrated and possibly being a good option for games.

Acknowledgment

The authors would like to thank CAPES and CNPq for their financial support.

References

[1] D. Lee, D. Lin, C.-P. Bezemer, and A. E. Hassan, "Building the perfect game–an empirical study of game modifications," *Empirical Software Engineering*, pp. 1–34, 2020.

[2] S. Pashkov, "Video game industry market analysis: Approaches that resulted in industry success and high demand," *Unit International Business Administration*, 2021.

[3] wepc, "Video Game Industry Statistics In 2020," https://www.wepc.com/news/video-game-statistics/, online; accessed 10 January 2021.

[4] E. M. Reyno and J. Á. C. Cubel, "Model driven game development: 2d platform game prototyping." *GAMEON*, pp. 5–7, 2008.

[5] K. Bilińska-Reformat, A. Dewalska-Opitek, and M. Hofman-Kohlmeyer, "To mod or not to mod—an empirical study on game modding as customer value co-creation," *Sustainability*, vol. 12, no. 21, p. 9014, 2020.

[6] P. Charles, R. M. Fuhrer, S. M. Sutton Jr, E. Duesterwald, and J. Vinju, "Accelerating the creation of customized, language-specific ides in eclipse," *ACM Sigplan Notices*, vol. 44, no. 10, pp. 191–206, 2009.

[7] H.-E. Eriksson and M. Penker, "Business modeling with uml," *New York*, pp. 1–12, 2000.

[8] F. Hamieh and B. Said, "Modeling and code generation of serious games for assessment," *ICT in our Lives, Alexandria, Egypt*, p. 8, 2018.

[9] K. K. Wollard and B. Shuck, "Antecedents to employee engagement: A structured review of the literature," *Advances in Developing Human Resources*, vol. 13, no. 4, pp. 429–446, 2011.

[10] R. C. Motta, K. M. de Oliveira, and G. H. Travassos, "Characterizing interoperability in context-aware software systems," *2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 203–208, 2016.

[11] M. J. B. Stephenson, J. Renz, X. Ge, and P. Zhang, "Generating stable, building block structures from sketches," *IEEE Transactions on Games*, pp. 1–10, 2019.

[12] E. S. de Lima, F. J. Gheno, and A. Viseu, "Sketch-based interaction for planning-based interactive storytelling," *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 154–162, 2020.

[13] F. Hamiye, B. Said, and B. Serhan, "A framework for the development of serious games for assessment," *International Conference on Games and Learning Alliance*, pp. 407–416, 2019.

[14] R. A. Boyd and S. E. Barbosa, "Reinforcement learning for all: An implementation using unreal engine blueprint," *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 787–792, 2017.

[15] S. Tang and M. Hanneghan, "A model-driven framework to support development of serious games for game-based learning," *Developments in E-systems Engineering*, pp. 95–100, 2010.

[16] K. Sánchez, K. Garcés, and R. Casallas, "A dsl for rapid prototyping of cross-platform tower defense games," *2015 10th Computing Colombian Conference (10CCC)*, pp. 93–99, 2015.

[17] A. W. Furtado, A. L. Santos, G. L. Ramalho, and E. S. de Almeida, "Improving digital game development with software product lines," *IEEE software*, vol. 28, no. 5, pp. 30–37, 2011.

[18] M. Petticrew and H. Roberts, *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008.

[19] S. Matalonga, F. Rodrigues, and G. H. Travassos, "Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review," *Journal of Systems and Software*, vol. 131, pp. 1–21, 2017.

[20] A. Matallaoui, P. Herzig, and R. Zarnekow, "Model-driven serious game development integration of the gamification modeling language gaml with unity," *2015 48th Hawaii International Conference on System Sciences*, pp. 643–651, 2015.

[21] N. Valcasara, *Unreal engine game development blueprints*. Packt Publishing Ltd, 2015.

[22] C. W. Krueger, "Software reuse," *ACM Computing Surveys (CSUR)*, vol. 24, no. 2, pp. 131–183, 1992.