An Evolutionary Approach for Generating Software Models: The case of Kromaia in Game Software Engineering

Daniel Blasco*, Jaime Font, Mar Zamorano, Carlos Cetina

Universidad San Jorge. SVIT Research Group
Autovía A-23 Zaragoza-Huesca Km.299, 50830, Zaragoza, Spain

Abstract

In the context of Model-Driven Engineering applied to video games, software models are high-level abstractions that represent source code implementations of varied content such as the stages of the game, vehicles, or enemy entities (e.g., final bosses).

In this work, we present our Evolutionary Model Generation (EMoGen) approach to generate software models that are comparable in quality to the models created by human developers. Our approach is based on an evolution (mutation and crossover) and assessment cycle to generate the software models. We evaluated the software models generated by EMoGen in the Kromaia video game, which is a commercial video game released on Steam and PlayStation 4. Each model generated by EMoGen has more than 1000 model elements.

The results, which compare the software models generated by our approach and those generated by the developers, show that our approach achieves results that are comparable to the ones created manually by the developers in the retail and digital versions of the video game case study. However, our approach only takes five hours of unattended time in comparison to ten months of work by the developers. We perform a statistical analysis, and we make an implementation of EMoGen readily available.

Keywords: Model-Driven Engineering, Search-based Software Engineering, Game Software Engineering

1. Introduction

Game Software Engineering (GSE) is a research area that was compared with classic Software Engineering for the first time by McShaffry in 2003 [1]. A recent survey, published in 2010, showed an overview of GSE research works and described the increasing interest in GSE [2]. Until now, GSE works have focused on issues like Requirement Traceability, but, due to the newness of this area, there are fields of study that remain unexplored.

As the survey mentioned above showed, Model-Driven Engineering (MDE) applied to video games is uncommon and, in general, is focused on generating source code from pre-existing models [3]. In the following years, subsequent works have continued focusing on the generation of source code from models [4, 5, 6, 7].

Some of the above MDE works [3, 7] use UML as the modelling language, one MDE work [6] uses process models, while the rest of them [4, 5] use Domain-Specific Modelling Languages (DSL). The major ad-

*Corresponding author. Tel.: +34 976060100
Email addresses: dblasco@usj.es (Daniel Blasco), jfont@usj.es (Jaime Font), mzamorano@usj.es (Mar Zamorano), ccetina@usj.es (Carlos Cetina)

vantage of modelling languages is that models use concepts that are much less bound to the underlying implementation technology, like video game engines such as Unreal [8] or Unity [9], and are much closer to the problem domain (the content of the video game) related to most popular programming languages (e.g., C++) [10]. This notion of "model" should not be confused with "mesh" or "polygon mesh", which are terms used in computer graphics and video games for the visual representation of 3D shapes/geometry.

In this work, we present our Evolutionary Model Generation (EMoGen) approach to generate software models that are comparable in quality to the models created by human developers. Automatically generating human-competitive software models is a challenging task. Fully achieving it spans the creation of model elements, the initialization of their properties, and their relationships with each other. Moreover, the resulting models must be valid, which includes satisfying modeling constraints. Finally, the human-competitive aspect is only achieved if the resulting models are comparable to those produced by software engineers for the same task at hand.

To generate the software models, our EMoGen approach takes an initial population of software models as input. These initial models may be randomly generated or may also be models that were previously generated by software engineers. Then, the genetic operations of EMoGen, which are mutation and crossover, evolve the population. Invalid models are fixed by means of repair operations. The evolved models are assessed by means of a fitness function. This evolution and assessment cycle is repeated until a stop condition is met. The output of EMoGen is a ranking of generated models.

The case study for our work are the game characters at the end of each stage of the video game Kromaia: final bosses. This video game was released worldwide in both physical and digital versions for PC and PlayStation 4. In the context of video games, bosses are particularly powerful adversaries that are generally much stronger than the rest of the enemies in the video game. Usually, the player must overcome them at the end of a stage or level. Three-dimensional space simulation titles, such as the case study, include content like: a spaceship controlled by a human player; architecture, buildings or celestial bodies; a repertory of bosses and basic enemies; and projectiles that are fired by both the human player and the enemies.

Figure 1 shows this game content in the context of a Kromaia playing session. Each of the stages or levels involves flying from a starting point to a certain destination, and the player spaceship must reach the goal before being destroyed. The stage implies exploring floating structures, avoiding asteroids and finding items along the route, which is protected by basic enemies (Figure 1, G): ships and creatures that try to damage the player unit by firing projectiles that damage the player spaceship (Figure 1, E). Basic enemies vary in anatomy and weaponry, but are significantly weaker than the player spaceship, in terms of endurance and firepower. In case that the player manages to reach the destination, the final boss (Figure 1, A) corresponding that stage appears, and it must be defeated in order to complete the stage. Bosses differ from basic enemies in some aspects:

- They are huge in comparison to the player spaceship or basic enemies. An average boss is 50 times larger than the player spaceship.
- Bosses tend to be more complex in terms of anatomical structure. They include mobile parts, and the nexuses that connect two parts may behave as joints or even strings.
- They use several powerful weapons that are distributed along the parts that form their structure or body. In contrast, basic enemies use one or two frontal weapons.
- Bosses include in their structure special parts, which are the only damageable elements of their bodies. The player must locate these parts and destroy them, since it is the only way to defeat a boss.

The final bosses of the video game Kromaia are specified with the Shooter Definition Model Language (SDML). SDML is a DSL model for the video game domain. Specifically, SDML defines aspects included in video game entities:

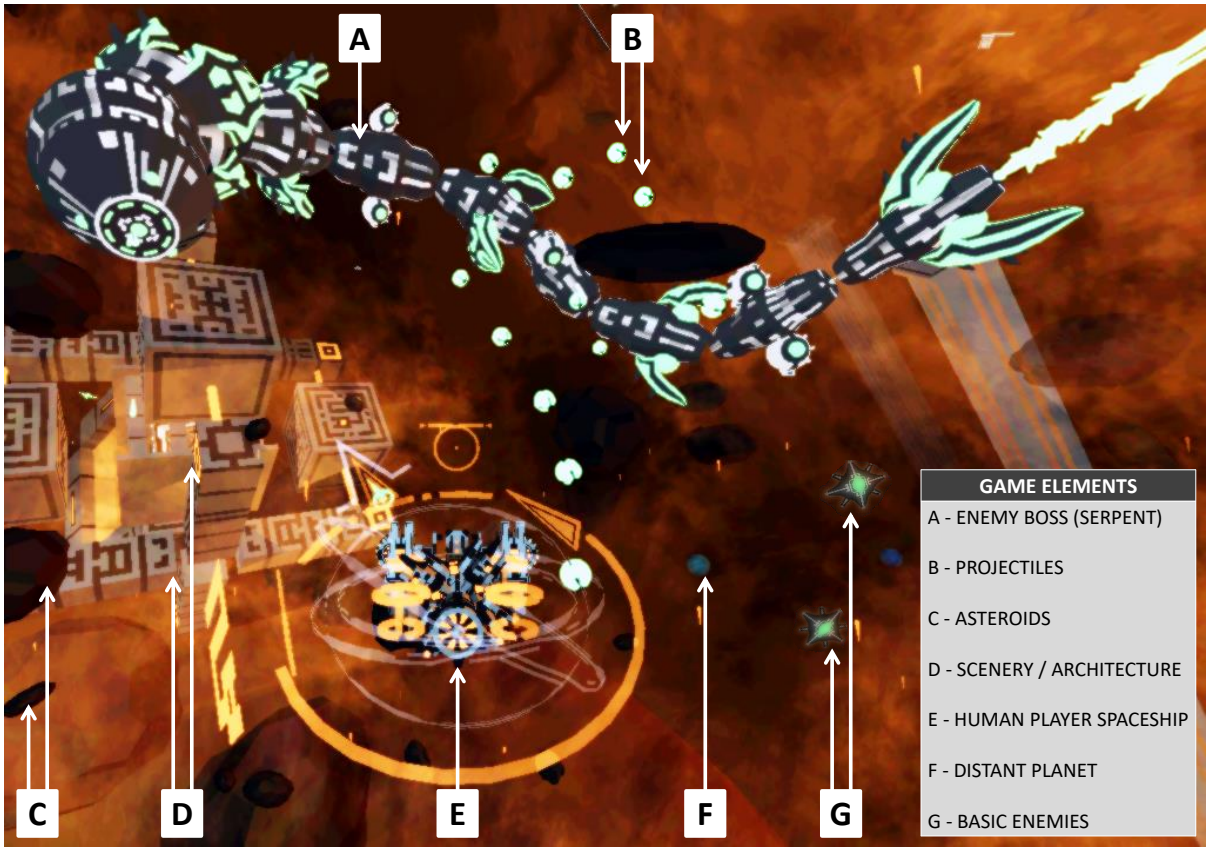

Figure 1: Screenshot showing game content in Kromaia.

- The anatomical structure, including which parts are used in it, their physical properties, and how they are connected to each other.
- The amount and distribution of vulnerable parts, weapons, and defenses in the structure/body of the character.
- The movement behaviours associated to the whole body or its parts.

This modeling language has concepts such as hulls, links, weak points, weapons, and AI components. The top of Figure 2 depicts an excerpt of the SDML that specifies one of the bosses of Kromaia (see the bottom of Figure 2). Each model element (e.g., Hull Head) instantiates a concept (e.g., Hull) of the modeling language in order to specify the boss. More can be learned about the SDML model of Figure 2 in the following video: `https://youtu.be/Vp3Zt4qXkoY`

Our evaluation considers different starting points for our approach (the initial population of software models), ranging from randomly initialized models to models generated by software engineers. Models generated by software engineers are the most promising starting points; however, they come at a cost for software engineers. In contrast, random models reverse the pros and cons. The baseline is Random Search, which in the past [11] has proven to outperform more sophisticated algorithms, and is a common sanity check practice in the Search-based Software Engineering [12] community. To evaluate the results,

we use measurements studied in the scientific literature of video games: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. We also perform a statistical analysis to provide evidence of the significance of the results.

The results show that our approach produces human-competitive software models for the content (final bosses) of a commercial video game. On average, these software models have about 1300 model elements. They can be produced in five hours of unattended time, which is a significant reduction in time compared to ten months of work by the video game developers, as the Kromaia's Version Control System shows [1]. What is specially relevant is that these human-competitive models are achieved in the most positive scenario for software engineers: the seeds are random models. This means that software engineers do not have to manually generate the initial population of software models.

This is a step forward for Genetic Programming [13], where programs are evolved to fit a specific task, in the context of MDE. This paper contributes to the rise of what we call Genetic Modeling, and in this case, models are evolved for video game content in the particular case of our evaluation. Furthermore, this acceleration of video game content generation is also relevant for software developers of video games since they face the challenge of what is called the age of crunch [14]. There is an ever-increasing high demand for game content that is derived from early access releases, post-launch updates, downloadable content, and games as a service.

We make an open-source implementation[2] of EMoGen available as well as two model examples to facilitate the reproduction of the results. Even though this implementation is adapted to Kromaia, our approach includes general ideas that could work in other domains, and therefore make EMoGen useful for encouraging Genetic Modeling.

The structure used in this paper is the following: Section 2 summarizes related works. Section 3 presents Model-Driven Engineering for Video Games. Section 4 describes our EMoGen approach. Section 5 deals with evaluation. Section 6 presents the discussion. Section 7 describes the threats to validity, and Section 8 presents the conclusion of the paper.

2. Related Work

This work is about generating software models using our EMoGen approach. Our evaluation is in the context of the video game content (bosses) of Kromaia. Therefore, our EMoGen approach generates models of Kromaia bosses. In this section, we discuss: 1) works that address game software engineering from the MDE community; and 2) works that address video game content generation. Video game content generation is also know as procedural content generation in the literature. Finally, we present an analysis of the research gap.

2.1. MDE and Game Software Engineering

Platform independence is one of the potential benefits of using models as the main artifact for software development. The diversity of platforms that video game developers must deal with has motivated most of the research works that combine software models and the domain of video games.

The 2010 survey of Software Engineering Research for Computer Games [2] identified only one work that applied Model-Driven Development to video games [3]. That work coined the term "Model-Driven Game Development" and presented a first approach to 2D platform game [3] prototyping through Model-Driven Development. Specifically, they used UML classes and state diagrams that were extended with stereotypes, and a model-to-code transformation to generate C++ code.

The research by Núñez et al. [4, 5] presents model-driven approaches that are intended to minimize errors, time, and cost in multi-platform video game development. The work in [4] proposes a

[1] Confirmed by the developers: two hours per day, including the time spent by real players on tests.

[2] `https://bitbucket.org/svitusj/EMoGen`

[3] One of the first genres in video game history. In platform games, the main character climbs and jumps between suspended platforms while avoiding enemies/obstacles.

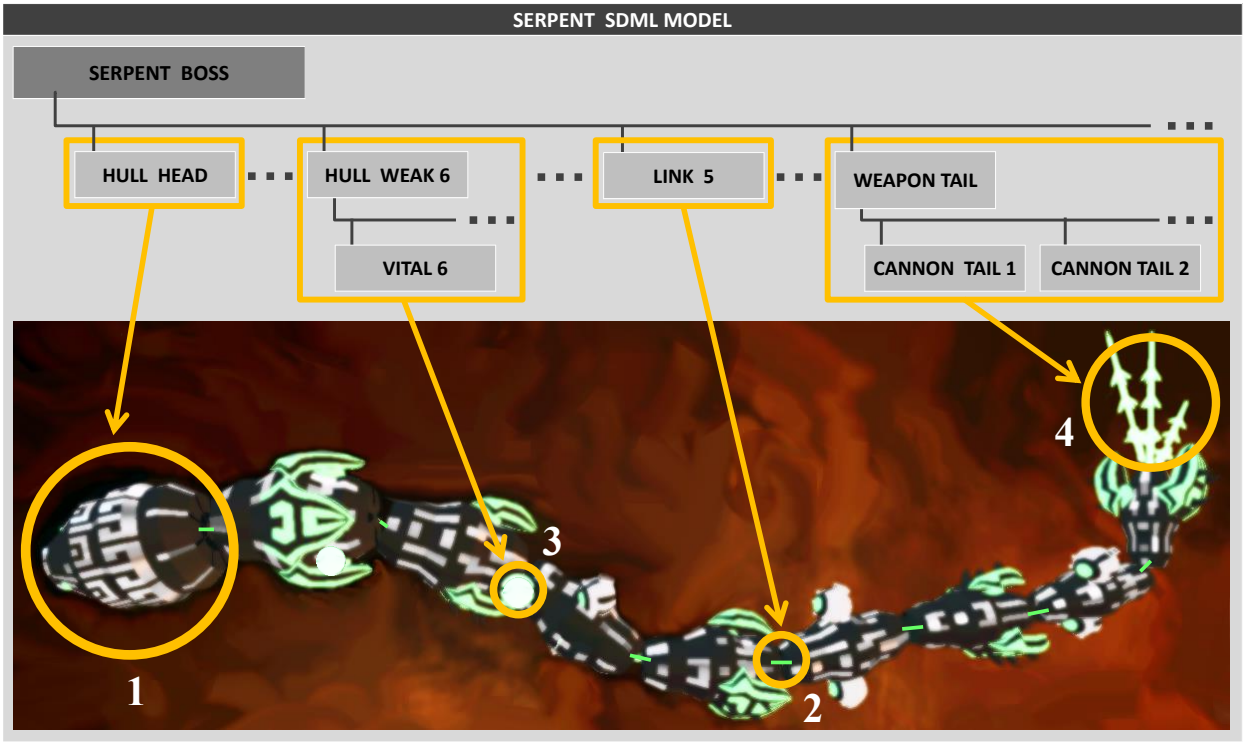

Figure 2: SDML model of a boss (top) and the boss at run-time (bottom).

Domains-Specific Language, named Gade4all, and focuses on tablet and smartphone-oriented games. Solís-Martínez et al. [6] suggest the use of business process models as the modeling language for video games. Specifically, they focus on the logic behind game loops in mobile games. In another work, Usman et al. [7] propose a model-driven product-line approach that focuses on multi-platform (Android and Windows Phone) mobile game development and maintenance. They use a feature model to configure the UML use-case, class, and state machine diagrams.

Although the details are different, the above works share a common assertion: in the domain of video games, automated code generation from software models has the potential to significantly reduce the development effort and cost. Paradoxically, platform independence is an issue that is being addressed by widely used technologies such as Unity [9] and Unreal Engine [8] and is leading developers to be less concerned with this issue in this particular domain.

In the intersection between software models and evolutionary computation, Williams et al. [15] use an evolutionary algorithm to search for desirable game character behaviours in a text-based video game that plays unattended combats and that outputs an outcome result. The character behaviour is defined using a Domain-Specific Language. The combats are managed internally and are only driven by behaviour parameters, without taking into account a spatial environment, real-time representation, or visual feedback (which takes into consideration the physical interaction of the characters, variation in the properties, etc.). However, the case study is a simplified text game. In addition, [15] deals with game parameter adjustment, that is, the work does not address the generation of software models.

Another work that focuses on the intersection between software models and evolutionary computation

is Avida-MDE [16], which generates state machines that describe the behaviour of one of the classes of a software system (Adaptive Flood Warning System case study). The resulting state machines comply with developer requirements (scenarios for adaptation). Instead of generating whole models, Avida-MDE extends already existing models (object models and state machines) with new state machines that support new scenarios. The work in [16] does not report the size of the generated state machines; however, the ones shown in the paper are around 50 model elements, which is significantly smaller than the more than 1000 model elements of the models of a commercial video game such as Kromaia.

2.2. Procedural Content Generation

Figure 3 shows the works of the video game research community that address procedural content generation (PCG). All of these works generate part of the content of video games using either evolutionary computation (15 of 28) or machine learning (8 of 28). They generate content for the following parts of games.

Game rules [17, 18, 19, 20, 21]. These are the core of the game and changing them could result in a new game. To generate game rules, research works combine rules from existing games, such as Checkers or Pac-Man. The results of these works are mainly obtained at the scale of board or grid-based games.

Level Layouts. These are generated by combining different pre-existing design elements of levels, such as terrain, platforms, items, non-player characters. Research works achieve results at the scale of games such as a clone of Super Mario Bros, which is adapted for research, or Quake, a shooter game. [22, 23, 24, 25, 26, 27, 28, 29, 30, 31]

Scenarios. These cover the structure of both puzzles [32, 33, 34] and maps/terrains [35, 36, 37, 38, 39]. These research works have been applied in grid-based puzzles. In the case of maps/terrains, these works have been applied in grid-based maps and heightmap terrains.

Items [40, 41, 42, 43, 44] include content such as weapons or buildings. Items are mostly generated by varying the properties of the items themselves. Research works create similar, but different, items in order to enrich players' experience. These works achieve results at the scale of games such as Galactic Arms Race, which is an indie development.

Furthermore, in Figure 3, we classify the works in relation to their type of assessment following the Togelius classification. Togelius et al. [45] classified assessment as direct, simulation-based, or interactive. Direct assessment is depicted as a red square in Figure 3 and uses features from the generated content to obtain a fitness value. Simulation-based assessment is depicted as a blue triangle and is based on artificial agents playing part of the game to evaluate the content. Finally, interactive assessment, which is depicted as an orange circle, involves the participation of real players scoring their gameplay explicitly or implicitly. The works with no defined assessment criteria could not be classified because they are surveys [27, 46] or they are not explained [20, 43]. They are represented in Figure 3 without a geometric classification.

2.3. Analysis of the research gap

There is a trend at the intersection of MDE and game software engineering (see subsection 2.1) where research works focus on achieving platform independence by means of the abstraction of software models. These works propose automation to generate the implementation code for different platforms from the models. However, none of these works have explored the generation of software models in the context of video games. In that context, generating software models results in generating game content.

In the video game research community (see subsection 2.2), research works explicitly address the generation of game content. So far, these works have succeeded in varying properties (works on items) and recombining pre-existing assets (works on game rules, level layouts, and scenarios). However, none of the works have leveraged models to generate game content.

Our work explores the gap of generating game content by leveraging software models. Our EMoGen approach evolves models and guides the evolution with a fitness function that uses the model interpreter for validation purposes and a game simulation which includes domain knowledge regarding the rules, the me-

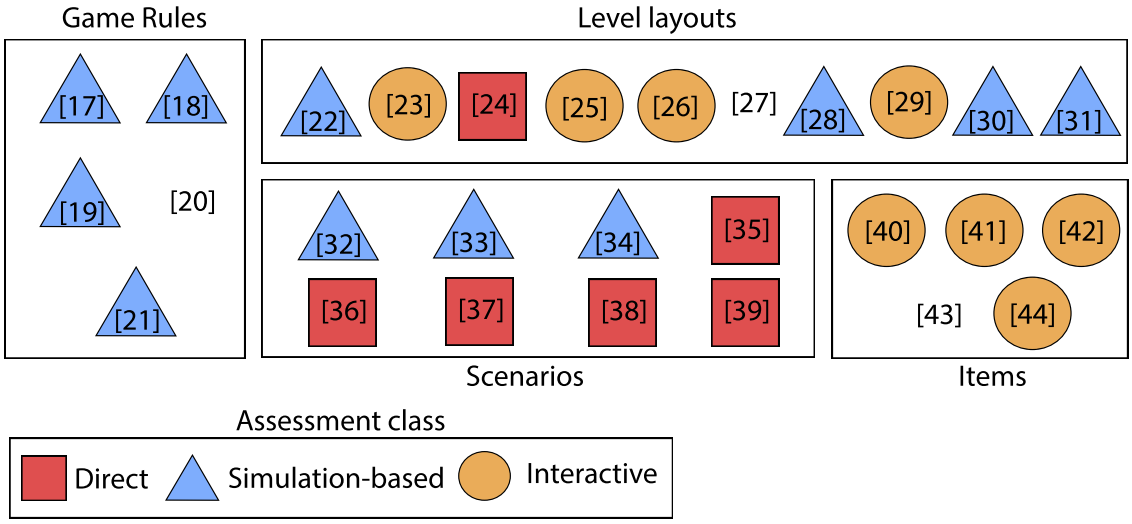

Figure 3: Overview of the PCG related work.

chanics, and the course of a playing session. Leveraging a model interpreter to generate content is one of the differences between our work and the previous works.

Our work achieves successful results at the scale of contemporary video games such as Kromaia, while none of the previous works address contemporary games, and most of them address academic mobile games. Compared to content generation works, our work addresses a different part of games: final bosses. Our work is not limited to varying properties as previous works on item generation. Furthermore, our work could be applied in settings where previous assets are not available: works on game rules, level layouts, and scenarios require the existence of previous assets.

3. Model-Driven Engineering for Video Games: The Kromaia Case

This section gives background on the role of models in Kromaia. Model-Driven Engineering (MDE) [47] aims to facilitate the development of complex systems by using models as the cornerstone of the software development process. Models are built in accordance with a metamodel that embodies the particularities and rules of a specific domain, formalizing what is valid and what is not when building a model for that metamodel. Models are used to formalize a system and capture each of its particularities. Then, those models can be used to reason about the system, perform validations, or transform it into different metalanguages, source code, or even run-time objects.

In the case of Kromaia, models are built against the Shooter Definition Model Language (SDML), a Domain-Specific Language created by Kraken Empire, which is the company that developed Kromaia. SDML allows for the definition of every element that will be present in the game, including worlds, vehicles, creatures, missions, enemies, etc. SDML is built using Ecore, the reference implementation of the Essential Meta Object Facility (EMOF) [48], which is the standard metalanguage proposed by the Object Management Group (OMG) to build metamodels.

Kromaia was developed with a custom video game engine, created by the company, that acts as a framework in the context of the video game architecture, as shown in Figure 4. This framework allows the developers to add new content in two different ways:

- **Programming**, making use of the Application Programming Interface (API) provided by the

framework.

- **Software Models**, which are created using SDML and translated to its programming equivalent at run-time by an interpreter that is used by the engine (shown in Figure 4).

The bottom of Figure 2 shows a final boss that is included in the video game case study. Examples of the SDML concepts used in bosses are the following:

Hulls and Links: Hulls (see circle 1 of Figure 2) are rigid bodies or solid objects that shape the structure of entities such as bosses. Hulls are connected via configurable nexuses, called links (circle 2 of Figure 2). Hulls and links define both the anatomical hierarchy and physics for the boss. Through different arrangement and flexibility settings, links determine whether a boss includes mobile structures, rigid parts, or even complex limbs that resemble tentacles.

Weak Points: Weak points (circle 3 of Figure 2) are concepts that are characterized by the fact that they can be damaged. Weak points are attached to hulls and they could optionally be arranged in layers to be unlocked as the player, who is the opponent of the boss, destroys them.

Weapons: Weapons (circle 4 of Figure 2) are objects that could inflict damage on direct contact, firing bullets, launching smart homing projectiles, or tracing rays/beams. These four kinds correspond to the weapon types used in Kromaia. These weapons automatically aim at targets (human players) since they involve AI behaviours.

AI components: The behaviour patterns shown by bosses during a battle are defined by Artificial Intelligence components. These elements do not have a graphical representation, so they are not highlighted in Figure 2. An AI module included in a boss may involve one or more of these concepts, suiting different battle situations or describing flocking behaviours.

The creation of game content in Kromaia is performed across four different stages using the concepts of SDML: Creative Design, Spatial Organization, Behaviour Specification, and Equipment Balance.

Creative Design: This is out of the scope of this work. Creative Design considers decisions from an artistic point of view. Therefore, it also involves concept art, texturing, and color palette selection, since the design must adjust to the art direction. This Creative Design is mostly related to texture files, which are a few of the properties of some elements of SDML.

What our work does consider for the case study are the following technical stages of bosses that are addressed by means of SDML.

1 Spatial Organization: The specification for the anatomy that characterizes a boss is defined at this level. Spatial Organization produces a hierarchy that makes bosses resemble chains, trees, rings, quadrupeds, bipeds, and an unlimited range of structures that are similar to those examples or that are combinations of them. This specification makes the hierarchy possible since it includes the hull set that is present in the boss and the links that connect them, which may vary in nature and use (e.g., ropes or fixed joints).

2 Behaviour Specification: At this point, it is assumed that the spatial organization for the boss is complete since Behaviour Specification revolves around the means that the boss will use for moving between target locations, exploring the environment, chasing enemies, or dodging attacks. Therefore, anatomical constraints may not be compatible with certain behaviours. During this stage, the developers assign different artificial intelligence behaviours in order to match game experience needs and agility requirements.

3 Equipment Balance: The last stage focuses on weak point and weapon distribution. Both the user experience and the difficulty associated to the boss are heavily influenced by the inclusion of different defense/attack items and the hulls to which they are attached. In addition, the weapon and weak point distribution affects and limits the possible or even valid strategies that human players could adopt to try to defeat the boss.

Even without Creative Design, the generation of boss models poses a challenge that exceeds the capabilities of systematic approaches. A boss model, without considering Creative Design, requires more than 1000 model elements. In an optimistic scenario where properties are ignored and model elements can be enabled or disabled, the resulting search space has 2^{1000} different possibilities. Trying to assess every single model is not feasible, and, therefore, our

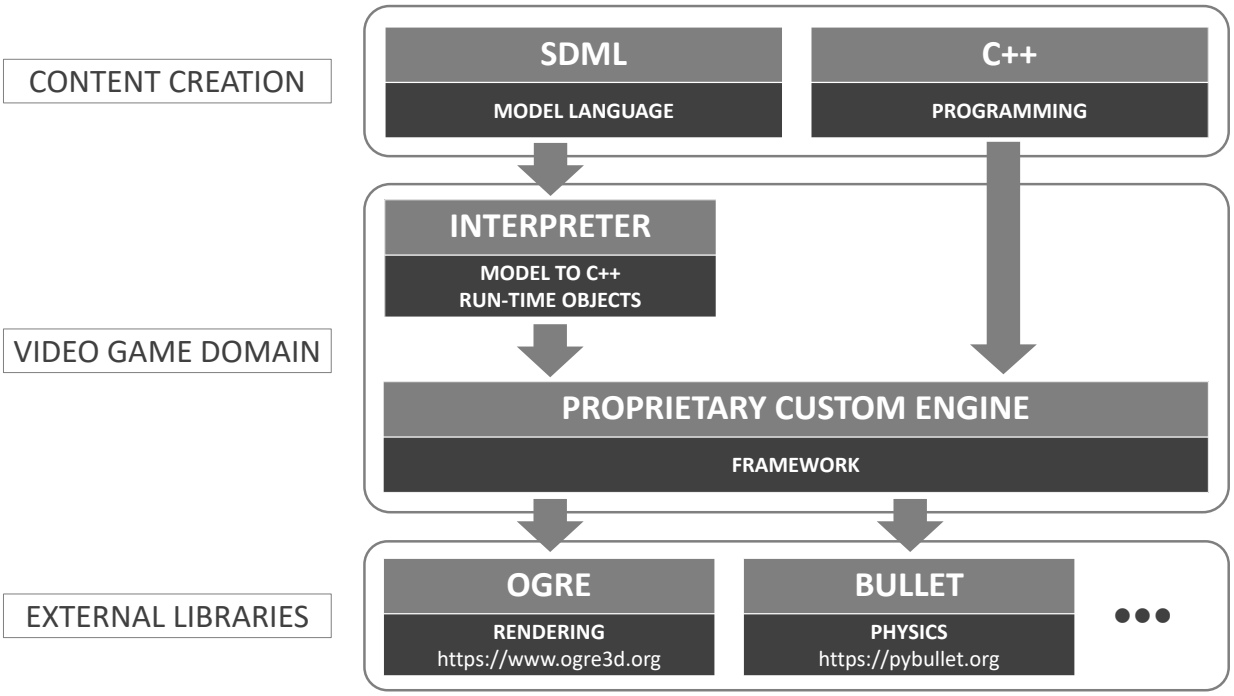

Figure 4: The different architecture layers in Kromaia, the video game case study.

EMoGen approach relies on an evolutionary algorithm to explore the search space.

4. Our EMoGen Approach

This section presents our EMoGen approach, which leverages evolutionary computation to generate human-competitive software models. Figure 5 shows the Evolutionary Algorithm (EA) that evolves a population of software models (models that follow our encoding) through genetic operations. First, the initial seeds are used to generate an initial population. Then, the population is assessed using the fitness function. Next, the population is evolved by applying genetic operators. This process (assessment + evolution) is repeated until the stop condition is met. Then, the population is decoded into models that are ready to be used.

When applying EMoGen to the Kromaia case study, we encode the models for the final bosses that are faced at the end of each level that are present in the video game. These models include the Spatial Organization, the Behaviour Specification, and the Equipment Balance of each of the final bosses (see Section 3).

4.1. Fitness of the EMoGen Approach

The objective of the fitness function in our EMoGen approach is to assess the quality of each individual as a model. This is done by taking into account the validity of the model and a game simulation that includes Domain Knowledge:

Validity: First, our approach checks the model in search of inconsistencies that would lead to classifying it as not being valid for use, hence assigning a fitness value of 0.

Domain Knowledge: Once the models are considered valid, our approach determines the suitability of the model. This is done by using a game simulation

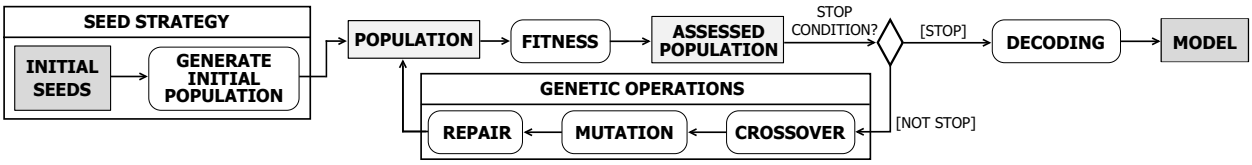

Figure 5: Overview of the EMoGen Approach.

that takes into account domain knowledge that is related to the elements present in a battle that involves a human player and a boss. It considers aspects such as the weaponry used by each of them and the differences between the two entities in terms of agility, speed, endurance, or size.

When applying EMoGen to the Kromaia case study, the validity of the models is performed by a run-time interpreter that is part of the game. The boss models generated by either human designers or our approach may not be valid due to inconsistent data that is related to the different stages (see Section 3). For instance, it is not valid to indicate in the model that a certain hull is connected to another hull that is not even present in that model. It is also invalid to denote a behaviour leading role for a hull that does not have at least one weapon attached. In these cases, the model would be assigned a fitness value of 0.

When no validation errors are found for a certain boss model and it is confirmed as valid, its fitness value is obtained from a simulation that reproduces a duel between the boss of the model and a human player. During that simulation, the player faces the boss in order to destroy the weak points that are available at that moment, whereas the boss acts according to the anatomy, behaviour, and attack/defense balance that is included in its model, trying to defeat the player. In that simulation, both the boss and the human player try to win the match and do not avoid confrontation, try to prevent draw/tie games, and try to ensure that there is a winner. The fitness value is calculated once the simulation process is finished, and our approach collects information on the battle progress and key events.

The information retrieved from the simulation is the data that the developers regard as relevant, using their domain knowledge, for determining whether or not a boss is suitable for a commercial release of the video game, i.e., the percentage of human player victories ($F_{Victory}$) and the percentage of human player health left once the player wins a duel (F_{Health}). The *clamp* function is used in the fitness measures:

$$clamp_{[0,1]}(x) = max(0, min(x, 1)) \tag{1}$$

In our approach, $F_{Victory}$ is calculated as a measure of the difference between the number of human player victories (V_P) and the optimal number of victories (33%, according to the developers of Kromaia and their criteria) ($V_{Optimal}$):

$$F_{Victory} = clamp_{[0,1]}\left(1 - \frac{\mid V_{Optimal} - V_P \mid}{V_{Optimal}}\right) \tag{2}$$

The criterion F_{Health}, which refers to completed duels that end in human player victories, is the average difference between the human player's health percentage once the duel is over (Θ_P) and the optimal health level that the player should have at that point ($\Theta_{Optimal}$, 20%, according to the developers):

$$F_{Health} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{V_P} \frac{|\Theta_{Optimal} - \Theta_P|}{\Theta_{Optimal}}}{V_P}\right) \tag{3}$$

$F_{Overall}$ is an average fitness value for a boss model that includes the fitness criteria described above and validation information, with $Validity$ being a value that determines whether or not a model is valid (1

and 0, respectively):

$$F_{Overall} = min(Validity, \frac{\sum_{i=1}^{N} F_i}{N}) \quad (4)$$

In the end, $F_{Overall}$ is a value in [0, 1] that is used to assess a boss model when our EMoGen approach is applied to the Kromaia case study.

4.2. Model Encoding of the EMoGen Approach

In evolutionary algorithms like the one used by our EMoGen approach, the models are encoded, and this representation is usually achieved in evolutionary algorithms with arrays or strings. In this work, we encode models elements in a way similar to our previous works [49, 50, 51] for models of the Induction Hob and Train Control domains.

When applying EMoGen to the Kromaia case study, the boss models contain elements, such as hulls and weapons, that are defined as being present or absent throughout the different stages in the creation process as well as properties that are constrained to a range of values. Figure 6 shows an excerpt the metamodel which the boss models are produced in accordance with. This excerpt omits secondary concepts, relationships and properties that are not as common or relevant as those presented in the figure. The metamodel contains more than 20 concepts, over 20 relationships and more than 60 properties. A final boss model like the example in Figure 2 and in the example video for this research (`https://youtu.be/Vp3Zt4qXkoY`) contains around 1300 elements.

Our approach encodes boss models as bi-dimensional matrices, in which columns correspond to the hulls that could be used in the model and in which each row indicates present or absent elements as well as properties:

Elements: Figure 7 shows four of the elements represented in our encoding. The presence or absence of these elements is defined through binary values (1 and 0, respectively). For instance, in the example shown in Figure 7, `Hull0` would not be present in the model and `Hull2` would have a turret.

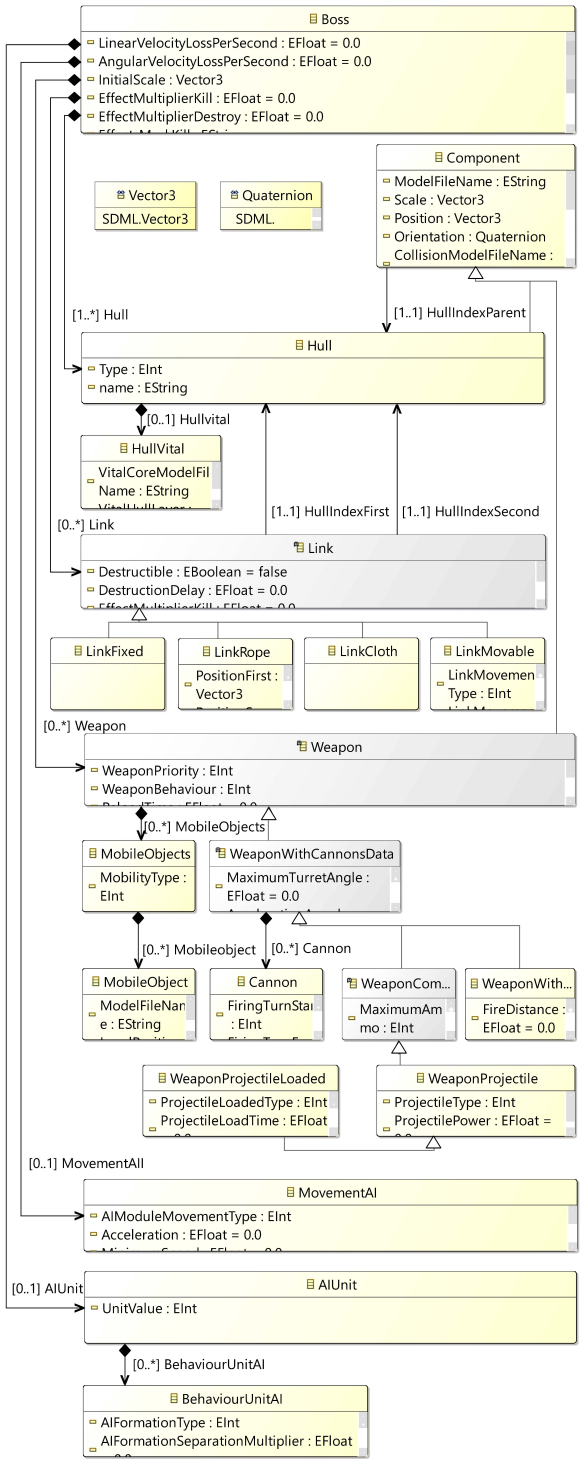

Figure 6: Excerpt of the Metamodel of the boss models.

ENCODING		HULLS				
		H 0	H 1	H 2	H 3	...
ELEMENTS AND PROPERTIES	ENABLED HULL	0	1	1	1	...
	LINK PARENT HULL	0	-1	4	2	...
	BEHAVIOUR LEAD	0	1	0	0	...
	GUN TURRET	1	0	1	1	...
	WEAK POINT	0	0	1	1	...
	...	...	...	...	...	...

Figure 7: Example of encoding for boss models.

Properties: The second row in Figure 7 shows a row with non-binary values that correspond to link relationships. This property indicates the parents that the hull is linked to. For instance, H3 of Figure 7 means that `Hull2` and `Hull3` are linked in a way that `Hull2` acts as a parent in the hierarchy. In addition, the encoding used by our approach represents hulls that do not depend on other hulls via links (root hulls) with values of -1 for that property, as shown in Figure 7.

4.3. Genetic Operations of the EMoGen Approach

Our EMoGen approach generates new models using some of the existing ones in the population as parents. This process is supported by genetic operators that are adapted to work with the EMoGen encoding that represents models.

First, it is necessary to select the parents from the model population before applying the genetic operators. The fittest of the potential parents are selected using the fitness value calculated for each model in the population.

Crossover: The crossover operation mixes the content of two models to create a new one. The new model takes a first random half with size n from the first parent and a second half with size S - n from the second parent, with S being the total size of the model.

Mutation: This operator is named after the mutations found in biology. These mutations make individuals show non-inherited modifications in their genes due to random factors. In our EMoGen approach, the mutation operation is applied on the new models that are created through crossover operations; however, changes depend on a certain probability, so they do not always occur. Due to the nature of the encoding used by our approach, mutations add and remove elements from the model and change properties.

Repair: Finally, after crossover and mutation have modified the genetic material of the individuals, inconsistencies may appear. For instance, when the crossover operation is applied, a link to a hull can be "broken", resulting in a new individual that is pointing to a hull that is not activated. Inconsistencies of this kind will prevent the model from being loaded into the game since it will fail to pass the model interpreter validation. The repair operator mitigates inconsistencies, making small modifications to the individuals, like modifying links that point nowhere. We do not claim to have a complete catalogue of repairs that guarantees that the resulting model will be accepted by the model interpreter.

5. Evaluation

This section presents the evaluation performed to determine if EMoGen can help game developers when creating the models for video games. In past works, there are four types of studies explained by Basili [52] and Travassos [53]. They refer to in-silico, in-vivo, in-vitro, and in-virtuo. More recent works used models as experimentation units [54] within in-virtuo experiments [55], but in this work we perform an in-silico experiment, in order to minimize the interaction with humans, and, therefore, favour the replicability of the study [53].

We defined the experimental design of the evaluation following the Goal-Question-Metric (GQM) [52] method. The GQM method defines a measurement model on three levels: a set of goals (the conceptual level) defined through a set of questions (the operational level) that can be answered through a set of metrics (the quantitative level). Following the template proposed by Basili et al. [56], we set three goals, which are defined through five questions that can be answered using seven metrics (see Figure 8). Apart from wall clock time, which is necessary to evaluate the time needed by our approach and the developers to generate models, a set of indicators of game quality described and widely used in the literature of

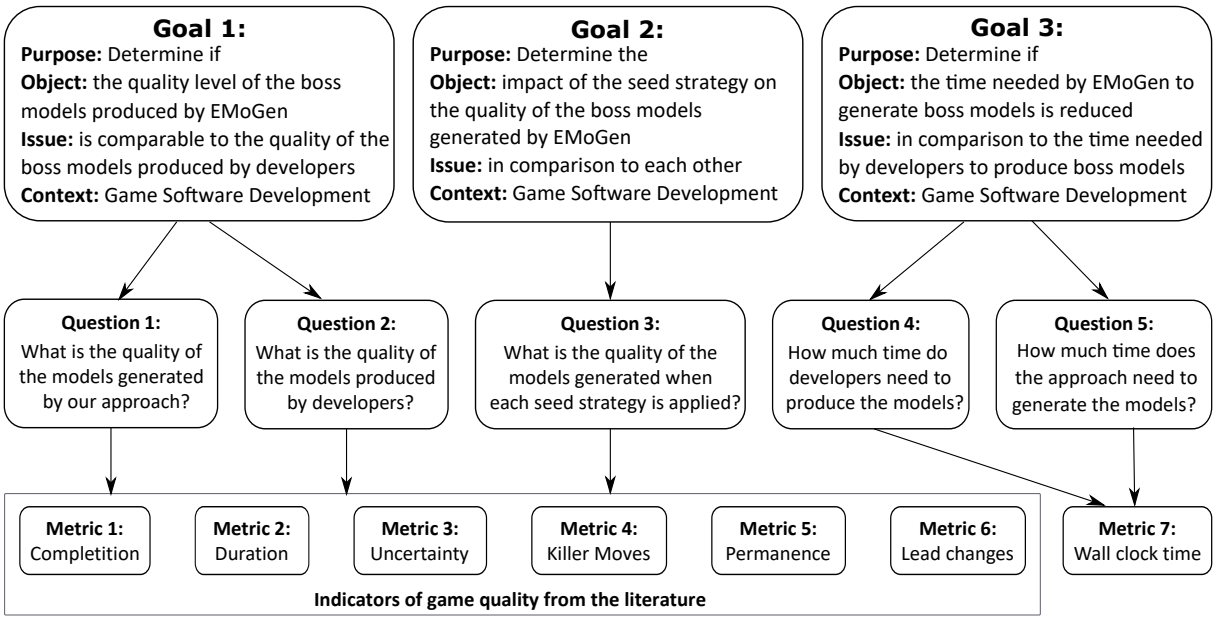

Figure 8: Application of the Goal Question Metric method for the evaluation

video game research are used to answer the questions. We use the six indicators that Browne et al. studied and recommended for being the most relevant [57]. Specifically, Browne et al. correlated 57 different quality indicators with players rankings. At the end, six of them stand out as the most important: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. Each of those metrics is measured using the characteristics of the case study. The suitability of a boss model is assessed studying the data obtained from a duel between the boss and a simulated player. That data provides values that are used in order to measure the metrics: the duration of a duel, the player victory percentage, the amount of relevant events in a match, and the health level of the player after the end of a duel.

Goal 1 is to determine if the quality level of the boss models produced by EMoGen is comparable to the quality of the boss models produced by developers. To determine this, we define two questions: Q1–What is the quality of the models generated by EMoGen?, and Q2–What is the quality of the models produced by developers?. The quality will be assessed using a set of six metrics that are widely used in the literature to assess the quality of games.

Goal 2 is to determine the impact of the seed strategies on the quality of the boss models generated by EMoGen in comparison to the quality of the boss models produced by the developers. The seeds are boss models that the evolutionary algorithm in our approach is fed with as starting points. To determine this impact, we define a new question: Q3–What is the quality of the models generated when each seed strategy is applied?. The quality level will be assessed as with Q1 and Q2, using six metrics from the literature.

Goal 3 is to determine if the time needed by EMoGen to generate boss models is reduced in comparison to the time needed by the developers to produce boss models. To determine this, we define two questions: Q4–How much time do developers need to produce the models? and Q5–How much time does EMoGen need to generate the models?. These two questions will be assessed by measuring the wall clock time.

The following subsections present a description of

the experimental setup, the metrics used to answer the questions, the details of the implementation, and the results.

5.1. Experimental Setup

Figure 9 shows an overview of the evaluation process followed. The first step of the evaluation is the extraction of information from the oracle that is provided by the developers of Kromaia (see top-left of Figure 9). The developers provided the set of final boss models and a set of seeds that is used by the approach to generate the initial population. Specifically, we use two types of seeds:

Final Boss: The type of boss that the player can find at the end of the level. Each final boss contains around 1300 model elements. The developers provided five different final bosses.

Miniboss: Enemies with less relevance in the game than a final boss but that are also built following the same stages and language (SDML). Each Miniboss contains around 500 model elements. The developers provided five different Minibosses.

Then, we perform a sanity check; we execute a random search to determine if the search space is large enough to benefit from the application of an evolutionary algorithm such as the one proposed here or a simple random search that is able to yield good results. To ensure a fair comparison, the random search is allocated with a budget that is similar to the one used by our approach. Specifically, the budget is in terms of the number of times the fitness function is executed as suggested in the literature [58].

Our approach is executed seven times, using a different seed strategy each time:

100R: The whole initial population is randomly generated so the seeds from the oracle are not used in this execution.

1F: A single final boss is randomly selected, out of the 5 available, and provided as initial seed. To generate the initial population, the seed is encoded as an individual and the rest of the population is obtained through the application of the mutation operation to the individual.

1M: A single Miniboss is randomly selected, out of the 5 available, and provided as initial seed. To generate the initial population, the seed is encoded as an individual and the rest of the population is obtained through the application of the mutation operation to the individual.

5F: The five final bosses are provided as initial seed. Similarly, the five final bosses are encoded as five individuals and the rest of the population is obtained through mutations of those five individuals.

5M: Similarly, the five Minibosses are provided as initial seed, encoded, and mutated to obtain more individuals and to complete the population.

95R+5F: The five final bosses are provided as initial seed. The five final bosses are encoded as individuals, but the rest of the population is randomly generated.

95R+5M: Similarly, the five Minibosses are provided as initial seed and encoded as individuals. The rest of the population is randomly generated.

As suggested in the literature [59], each execution of the approach is repeated 30 times to compensate for the stochastic nature of evolutionary algorithms. Then, the resulting boss models are measured using the six Quality measurements [60, 61, 62, 63, 64, 65, 66, 67, 68] (see the middle part of Figure 9). Similarly, the Boss Models obtained from the oracle are also subject to the same quality measurements. Then, all of the results are compared and statistically analyzed to determine the significance of the results.

In order to define the mathematical expressions that represent each of those quality measurements for the game studied, Kromaia, different tests and surveys were conducted with more than 30 users who belonged to the main target audience of the commercial case study.

The company responsible for the development of Kromaia provided data from their Version Control System with the help of the two engineers who produced and modified the boss models until the versions included in the final product were completed. These engineers have worked in the video game industry for 15 years and were informed of the purpose of the present work. Additionally, they signed a consent form before the data was used in this research. The company and these engineers collaborated in this work in order to research on possible improvements in the boss production process.

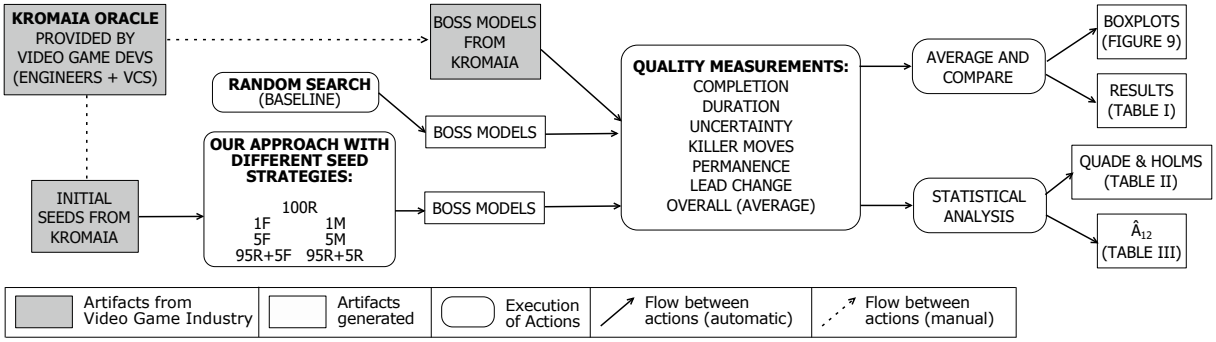

Figure 9: EMoGen Approach Evaluation Process.

5.2. Quality measurements

Previous research works have formalized fundamental and measurable indicators of game quality, like Depth and Decisiveness [60], Tension [64], Interestingness [61], Uncertainty [62], or Interaction [63]. In a more recent research done by Browne et al., the experimentation with game users showed that the following criteria stand out as being the most important: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change [57]. Our evaluation measures these criteria with values in the interval [0,1].

Completion (Viability): A game against a boss unit should end with more conclusions (victories for either the player or the boss) than draws/ties. The criterion $Q_{Completion}$ calculates a ratio of conclusions over total duel count:

$$Q_{Completion} = \frac{Conclusions}{Duels} \tag{5}$$

Duration (Viability): The duration of duels between players and boss units is expected to be around a certain optimal value. For the video game case study, through tests and questionnaires with players, the developers determined that concentration and engagement for an average boss reach their peak at approximately 10 minutes ($T_{Optimal}$), whereas the maximum accepted time was estimated to be 20 minutes ($2 * T_{Optimal}$). Significant deviations from that reference value are good design-flaw indicators: short games are probably too easy; and duels that go on a lot longer than expected tend to make players lose interest. The criterion $Q_{Duration}$ is a measure of the average difference between the duration of each duel (T_d) and the desired, optimal duration ($T_{Optimal}$):

$$Q_{Duration} = clamp_{[0,1]} \left(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels} \right) \tag{6}$$

Uncertainty (Quality): In order to keep players engaged with a duel, neither the player nor the boss unit should get extremely close to victory or defeat too early before the duel is settled, with (T_d) being its duration. Therefore, a duel is considered to be more uncertain the longer the time until the player's or the boss unit's health levels reach a dangerous/critical status (P_d and B_d, respectively). For each duel, $Q_{Uncertainty}$ measures the average deviation between the time at which it is detected that one of the contenders is on the verge of defeat and the time corresponding to the duration of the duel.

$$Q_{Uncertainty} = clamp_{[0,1]} \left(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min(P_d, B_d)}{T_d}}{Duels} \right) \tag{7}$$

Killer Moves: Q_{KMoves} measures the proportion of killer moves by any contender (K), taking into account the moves that are considered to be remarkable

highlights (H) but that are less important than killer moves. In the video game case study, the developers considered that a highlight move happens when either the boss unit or the player experiences a decrease in health; killer moves are those that make the difference in health between the contenders reach 30%.

$$Q_{KMoves} = clamp_{[0,1]} \left(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels} \right) \quad (8)$$

Permanence: Duels with a high permanence value are games in which the advantages given by significant actions or moves by one of the contenders are unlikely to be immediately reverted by the opponent in terms of dominance. In the video game case study, the developers considered every highlight move and killer move to be meaningful actions, with recovery moves (R) being those that quickly cancelled the advantages given by other previous killer or highlight moves. The criterion $Q_{Permanence}$ is measured as follows:

$$Q_{Permanence} = clamp_{[0,1]} \left(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d+K_d}}{Duels} \right) \quad (9)$$

Lead Change: The lack of lead changes indicates low dramatic value. In the video game case study, the lead is determined at any given moment by considering the contender with the highest health level. This criterion is measured taking into account those highlight or killer moves that cause the lead to change (L) during the course of a duel:

$$Q_{LChange} = clamp_{[0,1]} \left(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d+K_d}}{Duels} \right) \quad (10)$$

Our approach evaluated these six (N) criteria for each boss unit that is included in the commercial release of the video game case study in order to obtain a quality threshold that is useful for verifying whether the results obtained by our approach reach the same quality levels. $Q_{Overall}$ calculates an average quality value for a model, including all of the quality criterion studied:

$$Q_{Overall} = \frac{\sum_{i=1}^{N} Q_i}{N} \quad (11)$$

The above quality measure is used to determine how many of the models produced by our approach are comparable in quality to those present in the case study.

5.3. Implementation Details

In order to implement the approach, we used the TinyXML parser to process SDML models. In addition, the specifications of the computer used in the evaluation process were the following: Toshiba Satellite Pro L830 laptop, with an Intel® Core™ i5-3317U processor with 4GB RAM and Windows 8 64bit.

For the parameters of the EA, since the focus of this work is not the tuning of parameters, we used values from the literature that have proven to provide good results with models [49, 50, 51]. The mutation probability (p_m) depends on the number of hulls in the boss: `1/(Hulls Number)`.

In general, there are two atomic types of performance measures that are used to evaluate search algorithms: measures regarding speed and measures regarding quality of the solution. Since the focus of this paper is on the quality of the solution, we allocated a budget for each execution of the approach. Specifically, after running some prior tests, we identified the time of convergence, which is the point where the search reaches the peak and no further improvements occur, to be around 40 minutes of execution. To ensure convergence, the amount of wall clock time for each of the runs was set to one hour. A prototype of EMoGen can be found at: `https://bitbucket.org/svitusj/EMoGen`

5.4. Results:

Figure 10 shows the results of the execution of our approach for each of the seven combinations of

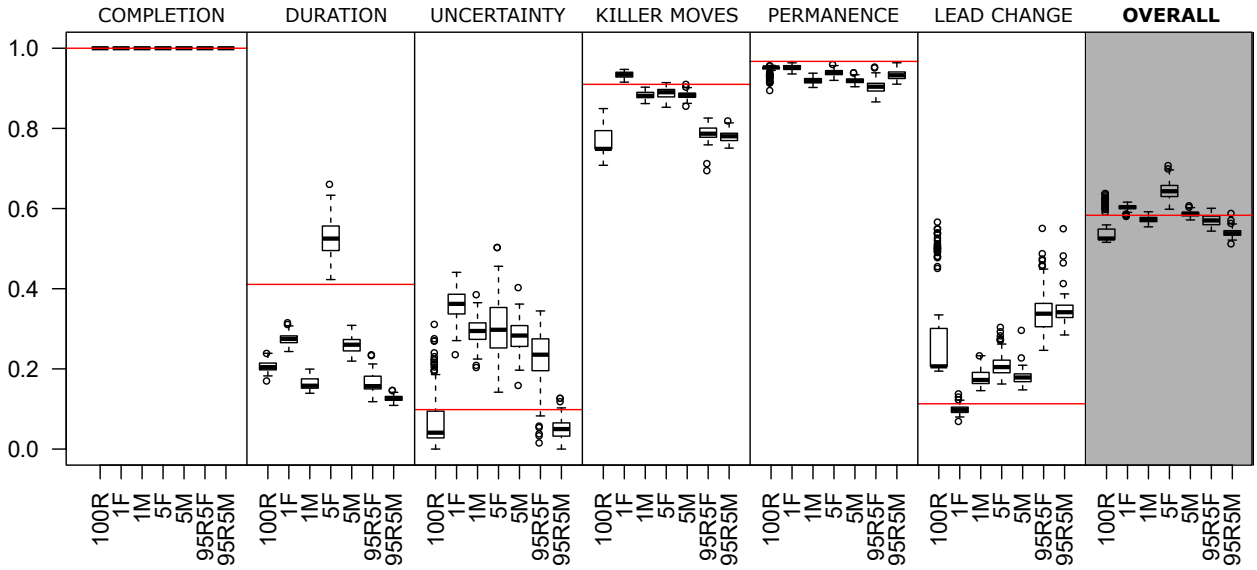

Figure 10: The results of the application of EMoGen with seven different seed strategies to generate final boss models. The results are grouped based on the 6 quality measures; the horizontal line in each group represents the value obtained by the original final boss models from Kromaia.

seeds and population strategy (100R, 1F, 1M, 5F, 5M, 95R+5F, 95R+5M). The executions are grouped to show the performance for a specific quality measurement (Completion, Duration, Uncertainty, Killer Moves, Permanence, Lead Change). The last column, with shaded background, shows the average of all of the quality measures for each execution. In addition, each population strategy for a specific quality is crossed by a horizontal line that indicates the value obtained by the human-generated final boss models that were obtained from the Kromaia oracle (see top-left of Figure 9).

Each boxplot is generated from the results of 30 executions [59] where each execution yields 100 individuals as a result. Therefore, each boxplot represents 3000 values of a specific quality in a final boss model. Figure 10 shows in each column how the quality values obtained for each of the seven strategies studied differ from the values for the models generated by the developers, which are represented by the horizontal lines that cross each column. The boxplots that are closer to the horizontal lines are more similar in quality to the models produced by the developers. Additionally, the use of boxplots allows for the representation of the different results for the strategies used.

Similarly, Table 1 shows the values obtained by each seed strategy (rows) and each of the quality measurements (columns). Each value is reported from 0 to 1, which are the worst and best possible values, respectively.

In addition, the first row shows the results for the sanity check: a Random search executed with a budget that is similar to our approach in terms of fitness executions (i.e., 3 million fitness executions). The purpose of the sanity check is to determine whether there is a need for a complex search strategy or the solutions to the problem can be found by mere chance. In our case, none of the individuals that were generated as part of the random search were able to be validated by our model interpreter; therefore, their score is 0.

The answer to Q1, which asks about the quality of the models generated by our approach, can be seen in the boxplots of Figure 10 and in Table 1: they show the values of each of the metrics for the different

seed strategies. Similarly, **the answer to Q2**, which asks about the quality of the models produced by developers, can be seen as the horizontal lines of Figure 10, which cross each column 10, and the associated values from Table 1 (last row, Kromaia Oracle). To address Goal 1, we compared the results obtained by our approach with those from the oracle. The values were similar, particularly in terms of the overall quality, with differences of around 5% at maximum. Therefore, we can conclude that the approach is able to generate final boss models that are comparable to those from Kromaia, the video game case study, whose final boss models were created manually by software engineers.

5.5. Statistical Analysis

To answer Q3 and to compare the impact of each of the seed strategies on the quality of the results, the empirical data was analyzed following the guidelines from the literature [69]. The statistical analysis included a significance test, the corresponding post-hoc analysis, and an effect size measure.

5.5.1. Statistical Significance

We applied a statistical test to the results of the seven seed strategies to determine if there were significant differences among the final boss models produced in terms of the quality measurements presented (i.e., the differences in the results were not obtained by mere chance).

After running the approaches a large enough number of times (30 as suggested by the literature [69]), we applied the Quade test since our data does not follow a normal distribution and the Quade test has proven [70] to be better than the rest of the non-parametric tests when working with real data.

The Quade test results in a p-value between 0 and 1, with 0 indicating that there are significant differences among the different seeds strategies and 1 indicating that there are no such differences. The threshold accepted by the research community is 0.05 [69], meaning that p-values below that number are statistically significant.

The results of the Quade test give a p-value below the 0.05 threshold for all of the measurements ($\ll 2.2x10^{-16}$), indicating that the differences observed in the results are significant enough to be caused by the seed strategy and are not due to mere chance. The test was not applied for the Completion quality measure since all of the results were 1 and there was no variance.

5.5.2. Post-hoc analysis

The Quade test only determined that there are differences among all of the seed strategies. To identify the specific seed strategies yielding significant differences, we applied a post-hoc analysis. This analysis consists of pair-wise comparisons of the results of each seed strategy to determine if there are statistically significant differences among the results of each pair of strategies.

We applied the Holm's post-hoc analysis, which is the most common post-hoc analysis applied after a Quade test [71]. Again to interpret the results, a value below 0.05 indicates that the differences between the two strategies are significant enough to be considered to be caused by the seed strategies. Table 2 shows the results of the post-hoc analysis. Each column shows the p-value for a specific quality measurement, while each row shows one of the pair-wise combinations of two seed strategies (the order does not matter for this test). Table 2 shows only the pairs of seed strategies that obtained values above 0.5, which cannot be considered significant enough to determine that the differences are due to the seed strategy. Pairs of strategies not shown in the table obtained values below 0.5 for all of the measurements, so the differences are due to the seed strategy applied.

For instance, the differences between 1M and 5M seed strategies cannot be considered significant enough for some of the measurements like uncertainty, killer moves, permanence, or lead change (see row labeled as 1M vs 5M in the middle of Table 2). It is common to obtain these results because the differences between some pairs of seed strategies are subtle.

These results partially answer Q3. Taking into account the differences in the results from Table 1 and the fact that they are significant as shown by the Holm's post-hoc analysis (see Table 2), we can conclude that the selection of the seed strategy does have

Table 1: Mean Values and Standard Deviations for each of the quality metrics (columns), and each of the seed strategies (rows). The seed strategies that achieve the best and worst results for each metric are highlighted in grey.

	COMPLETION	DURATION	UNCERTAINTY	KILLER MOVES	PERMANENCE	LEAD CHANGE	OVERALL
Random Search	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
100R	1 ± 0	0.21 ± 0.01	0.07 ± 0.08	0.77 ± 0.04	0.95 ± 0.01	0.28 ± 0.13	0.55 ± 0.04
1F	1 ± 0	0.28 ± 0.02	0.36 ± 0.04	0.93 ± 0.01	0.95 ± 0.01	0.10 ± 0.01	0.60 ± 0.01
1M	1 ± 0	0.16 ± 0.02	0.30 ± 0.03	0.88 ± 0.01	0.92 ± 0.01	0.18 ± 0.02	0.57 ± 0.01
5F	1 ± 0	0.53 ± 0.05	0.30 ± 0.07	0.89 ± 0.01	0.94 ± 0.01	0.21 ± 0.03	0.64 ± 0.02
5M	1 ± 0	0.26 ± 0.02	0.28 ± 0.04	0.88 ± 0.01	0.92 ± 0.01	0.18 ± 0.02	0.59 ± 0.01
95R+5F	1 ± 0	0.17 ± 0.02	0.23 ± 0.07	0.79 ± 0.02	0.90 ± 0.02	0.34 ± 0.05	0.57 ± 0.01
95R+5M	1 ± 0	0.13 ± 0.01	0.05 ± 0.03	0.78 ± 0.01	0.93 ± 0.01	0.35 ± 0.04	0.54 ± 0.01
Kromaia Oracle	1	0.41	0.10	0.91	0.97	0.11	0.58

Table 2: Holms Post Hoc $pValues$ for each quality metric (columns) and each pair of seed strategies (rows) whose value is above the threshold (0.05). Missing pairs of seed strategies obtain values below the threshold and are omitted for legibility.

	DURATION	UNCERTAINTY	KILLER MOVES	PERMANENCE	LEAD CHANGE	OVERALL
100R vs 1F	$\ll 2.2x10^{-16}$	$\ll 2.2x10^{-16}$	$\ll 2.2x10^{-16}$	0.116	$\ll 2.2x10^{-16}$	$\ll 2.2x10^{-16}$
100R vs 95R+5M	$\ll 2.2x10^{-16}$	0.36651	$1.3x10^{-6}$	$2.3x10^{-12}$	0.0033	1
1M vs 5F	$\ll 2.2x10^{-16}$	0.36651	0.092	$\ll 2.2x10^{-16}$	$2.1x10^{-9}$	$\ll 2.2x10^{-16}$
1M vs 5M	$\ll 2.2x10^{-16}$	0.36651	0.726	0.998	0.8621	$5.3x10^{-6}$
1M vs 95R+5F	0.367	$8.6x10^{-7}$	$\ll 2.2x10^{-16}$	$9.9x10^{-9}$	$\ll 2.2x10^{-16}$	1
95R+5F vs 95R+5M	$1.8x10^{-15}$	$8.8x10^{-16}$	0.187	$\ll 2.2x10^{-16}$	0.8621	$1.8x10^{-11}$

an impact on the quality of the final boss model produced in terms of the quality measurements included in this study.

5.5.3. Effect Size

It has been proven that statistically significant differences can be obtained even if they are so small as to be of no practical value [69]. To completely study Goal 2, we analyzed the effect size to determine the magnitude of the improvement of one seed strategy over the others. To do this, we measured the Vargha and Delaneyś $\hat{A}_{12}$ non-parametric effect size [72, 73]. $\hat{A}_{12}$ can be used to measure the probability of one seed strategy yielding better results than another one in terms of the quality measurements analyzed.

The $\hat{A}_{12}$ between a pair of seed strategies is expressed as a value between 0 and 1 and indicates the probability of the first seed strategy yielding better results than the second. Table 3 shows the $\hat{A}_{12}$ results for each pair of seed strategies and measurement (i.e., if the results showed significant differences in the Holm's test). Extreme values indicate where the higher differences reside and are highlighted. The values above 90% are shown in dark grey, and the values below 10% are highlighted in light grey.

For instance, the fourth row (Table 3) is labeled as 100R vs 5M, so each of the cells shows the percentage of times that applying the 100R seed strategy produces better results than applying the 5M seed strategy for a specific quality measurement. For instance, the value of Duration (first column) is 1.38%, so 100R yields better Duration values than 5M only 1.38% of the times. It is important to note that the values can be read both ways, so the 5M strategy produce better results than the 100R strategy for the Duration quality measurement 98.62% of the times.

To address Goal 2, which deals with determining the impact of the seed strategy on the quality of the boss models, we need **to answer Q3**, which asks about the quality of the models for each of the strategies studied: On average, the 5F seed strategy provides the best results (better than any other strategy 99% of the times), followed by the 1F strategy (around 80% of the times), followed by the 5M strategy (around 60% of the times), followed by 1M and 95R+5F (around 40% of the times), followed by the 100R (around 20% of the times) and the 95R+5M strategy (only around 12% of the times).

Table 3: The $\hat{A}_{12}$ statistic for each quality metric (columns) and pair of seed strategies (rows) with significant differences according to the Holms post hoc. Values above 90% are highlighted in dark grey and values below 10% are highlighted in light grey.

	DURATION	UNCERTAINTY	KILLER MOVES	PERMANENCE	LEAD CHANGE	OVERALL
100R vs 1M	98.6 %	1.68 %	0 %	95.09 %	93.88 %	23.04 %
100R vs 5F	0 %	3.16 %	0 %	77.11 %	65.7 %	3.04 %
100R vs 5M	1.38 %	2.74 %	0 %	95.19 %	96.26 %	22.68 %
100R vs 95R+5M	100 %	45.72 %	25.82 %	82.74 %	23.24 %	29.31 %
1F vs 1M	100 %	88.92 %	100 %	99.97 %	0 %	99.72 %
1F vs 5F	0 %	75.90 %	100 %	88.42 %	0 %	2.67 %
1F vs 5M	72.06 %	91.73 %	100 %	99.92 %	0 %	94.35 %
1F vs 95R+5F	100 %	96.79 %	100 %	98.75 %	0 %	98.39 %
1F vs 95R+5M	100 %	100 %	100 %	92.27 %	0 %	99.95 %
1M vs 95R+5M	99.69 %	100 %	100 %	15.29 %	0 %	98.26 %
5F vs 5M	100 %	59.51 %	65.72 %	96.26 %	84.82 %	99.86 %
5F vs 95R+5F	100 %	76.89 %	100 %	96.05 %	0.74 %	99.98 %
5F vs 95R+5M	100 %	100 %	100 %	68.26 %	0.06 %	100 %
5M vs 95R+5F	99.76 %	74.84 %	100 %	82.76 %	0.19 %	82.68 %
5M vs 95R+5M	100 %	100 %	100 %	14.71 %	0.02 %	99.27 %

To address Goal 3, which deals with determining if the time needed by the developers to generate boss models is reduced by our approach, we need **to answer Q4 and Q5**, which ask about the time that the developers and our approach need in order to produce boss models, respectively: It was necessary to analyze the Version Control System used by the developers to determine the time that was originally spent to build the five final boss models. The sum of the time spent in the three development stages involved in this study (Spatial Organization, Behaviour Specification, and Equipment Balance) that led to the original final bosses that were commercially released was 10 months. Our approach needed approximately one hour for the execution of each of the seed strategies. Therefore, to have a fair comparison, we would need to execute the EA five different times to generate five different final bosses, as in the original game, resulting in five hours. In other words, the approach is able to yield comparable results in less than a thousandth part of the original time required.

6. Discussion

Before conducting the experiment studied in this work, we thought that human-competitive results might be achievable by taking one or several final boss models as the starting point. This coincides with the idea of Genetic Improvement [74], for which the results are obtained using seeds that are similar enough to solutions. Paradoxically, the main disadvantage associated to using final boss models as seeds is that it is necessary to obtain those models in advance, and it is time-consuming for humans to generate such complete models. A random sample taken from the results suggested that using final boss models as seeds could lead to final bosses that are very similar to those seeds, which is another disadvantage if the final bosses only provide small variations instead of new, varied video game content that is found to be engaging and not repetitive by users.

We also used miniboss models as seeds. These models include over 500 model elements, whereas a final boss model could involve around 1300 model elements. Using various different miniboss models as seeds shows that the final boss models obtained include characteristics found in the seeds. These bosses, which are significantly less complex to design, could be useful for controlling the characteristics in the final boss models generated. However, we must study this possibility carefully in future works.

We also tested our approach with an initial population that consisted of models that were generated randomly. The combination of our fitness and re-

pair operations makes it possible for these models to evolve in order to achieve models that are comparable to those provided by the developers. This coincides with the idea of Genetic Programming [75], which generates a complete program using genetic encoding. In our case, this is Genetic Software Modeling since, in our work, we deal with software models. The results obtained in our work do not claim that it is possible to obtain a complete model of a whole video game with our EMoGen approach. However, our results do show that it is possible to perform Genetic Software Modeling to achieve results that are comparable to the bosses in the commercial releases of Kromaia. This is feasible because we apply our approach to models, which have less noise than source code because software models abstract from implementation details.

An issue to be addressed in the future is the use of our approach in other industrial contexts. Commercial engines, like Unity [9] or Unreal, which uses its own DSL named BluePrints [8], are widely adopted by development teams, and their architecture is similar to that shown in Figure 4. These DSLs are similar to SDML in terms of level of detail, and allow for the description of every element present in a game. Since the ideas proposed in this work are general, their application to different commercial DSLs is part of our future work.

7. Threats to validity

Following the guidelines suggested by De Oliveira et al. [76], we have identified the following threats to validity:

Not accounting for random variation: We addressed this threat by performing 30 runs for each of the executions of our approach.

Lack of a meaningful comparison baseline: We addressed this threat by comparing our approach with a random search and also by comparing the results with the final bosses from the developers of the commercial release of the video game case study.

Lack of clarity on data collection: We addressed this threat by using the data provided by the SDML models of the contenders to perform the simulation and two main indicators that were obtained from the developers and used to give value to configurations: victory percentage and health level.

Lack of real problem instances: The case study used in the evaluation is an industrial video game, and the problem artifacts were directly obtained from the video game industry.

Lack of assessing the validity of cost measures: We performed a fair comparison between the final bosses from Kromaia and the bosses generated by our approach, studying the time spent by the developers and our algorithms to obtain the results.

Lack of assessing effective measurements: We addressed this threat by using quality measures that are presented in the literature of video game research [57].

8. Conclusion

Our EMoGen approach produces content for video games, whole models of final bosses that could be used in Kromaia, the video game case study. The production of these models is relevant for the creation of the video game, and processes like updates or expansions, which are demanding in terms of quality and release schedule.

The quality of the bosses obtained by our approach is comparable to that achieved by the professional video game developers that produced the final bosses that were included in the commercial release of the case study.

The results show that the seeds used, the final boss models which the evolutionary algorithm of our approach is fed with as starting points, have an impact in the quality of the bosses produced: the use of the final bosses or minibosses included in the commercial video game case study helps our approach obtain boss models of higher quality in comparison to those produced when the seeds are random models.

EMoGen, which uses DSL models, propose ideas which do not make our approach depend on the video game studied in this work. Therefore, the applicability of our approach could be studied in the context of other commercial frameworks.

Our approach only takes five hours of unattended time in comparison with ten months of work by

the video game developers. Our work also offers a relevant result for genetic software modeling since human-competitive software models can even be achieved from randomly generated models, i.e., without a starting modeling effort from developers. We have made two model examples and an implementation of EMoGen freely available in order to facilitate the adoption of our approach.

Acknowledgments

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the Project ALPS (RTI2018-096411-B-I00).

References

[1] M. McShaffry, Game Coding Complete, Paraglyph Publishing, 2003.

[2] A. Ampatzoglou, I. Stamelos, Software engineering research for computer games: A systematic review, Information and Software Technology 52 (2010) 888 – 901.

[3] E. M. Reyno, J. A. Carsí Cubel, Automatic prototyping in model-driven game development, Comput. Entertain. 7 (2009) 29:1–29:9.

[4] E. R. Núñez-Valdéz, V. García-Díaz, J. M. C. Lovelle, Y. S. Achaerandio, R. G. Crespo, A model-driven approach to generate and deploy videogames on multiple platforms, J. Ambient Intelligence and Humanized Computing 8 (2017) 435–447.

[5] E. R. Núñez-Valdéz, O. S. Martínez, B. C. P. García-Bustelo, J. M. C. Lovelle, G. Infante-Hernandez, Gade4all: Developing multiplatform videogames based on domain specific languages and model driven engineering, IJIMAI 2 (2013) 33–42.

[6] J. Solís-Martínez, J. P. Espada, N. García-Menéndez, B. C. P. García-Bustelo, J. M. C. Lovelle, VGPM: using business process modeling for videogame modeling and code generation in multiple platforms, Computer Standards & Interfaces 42 (2015) 42–52.

[7] M. Usman, M. Z. Iqbal, M. U. Khan, A product-line model-driven engineering approach for generating feature-based mobile applications, Journal of Systems and Software 123 (2017) 1–32.

[8] E. Games, Unreal engine, version 2018.3.9, 1998. URL: `http://www.unrealengine.com/`.

[9] U. Technologies, Unity, version 2018.3.9, 2005. URL: `https://unity.com/`.

[10] B. Selic, The pragmatics of model-driven development, IEEE Software 20 (2003) 19–25.

[11] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.

[12] M. Harman, B. Jones, Search-based software engineering, Information Software Technology 43 (2001) 833–839.

[13] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.

[14] IGDA, International Game Developers Association, 2018. URL: `http://tiny.cc/ucev5y`.

[15] J. R. Williams, S. M. Poulding, L. M. Rose, R. F. Paige, F. A. C. Polack, Identifying desirable game character behaviours through the application of evolutionary algorithms to model-driven engineering metamodels, in: Search Based Software Engineering - Third International Symposium, SSBSE 2011, Szeged, Hungary, September 10-12, 2011. Proceedings, 2011, pp. 112–126. URL: `https://doi.org/10.1007/978-3-642-23716-4_13`. doi:`10.1007/978-3-642-23716-4\_13`.

[16] H. J. Goldsby, B. H. C. Cheng, Automatically generating behavioral models of adaptive systems to address uncertainty, in: K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, M. Völter (Eds.), Model Driven Engineering Languages and Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 568–583.

[17] V. Hom, J. Marks, Automatic design of balanced board games, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2007, pp. 25–30.

[18] J. Togelius, J. Schmidhuber, An experiment in automatic game design, in: 2008 IEEE Symposium On Computational Intelligence and Games, IEEE, 2008, pp. 111–118.

[19] C. B. Browne, Automatic generation and evaluation of recombination games, Ph.D. thesis, Queensland University of Technology, 2008.

[20] A. M. Smith, M. Mateas, Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, IEEE, 2010, pp. 273–280.

[21] C. Salge, T. Mahlmann, Relevant information as a formalised approach to evaluate game mechanics, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, IEEE, 2010, pp. 281–288.

[22] J. Togelius, R. De Nardi, S. M. Lucas, Making racing fun through player modeling and track evolution (2006).

[23] C. Pedersen, J. Togelius, G. N. Yannakakis, Modeling player experience in super mario bros, in: 2009 IEEE Symposium on Computational Intelligence and Games, IEEE, 2009, pp. 132–139.

[24] N. Sorenson, P. Pasquier, Towards a generic framework for automated video game level creation, in: European conference on the applications of evolutionary computation, Springer, 2010, pp. 131–140.

[25] M. Jennings-Teats, G. Smith, N. Wardrip-Fruin, Polymorph: A model for dynamic level generation, in: Sixth Artificial Intelligence and Interactive Digital Entertainment Conference, 2010.

[26] N. Sorenson, P. Pasquier, S. DiPaola, A generic approach to challenge modeling for the procedural creation of video game levels, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 229–244.

[27] R. Van Der Linden, R. Lopes, R. Bidarra, Procedural generation of dungeons, IEEE Transactions on Computational Intelligence and AI in Games 6 (2013) 78–89.

[28] J. Roberts, K. Chen, Learning-based procedural content generation, IEEE Transactions on Computational Intelligence and AI in Games 7 (2014) 88–101.

[29] W. L. Raffe, F. Zambetta, X. Li, K. O. Stanley, Integrated approach to personalized procedural map generation using evolutionary algorithms, IEEE Transactions on Computational Intelligence and AI in Games 7 (2014) 139–155.

[30] M. Nogueira-Collazo, C. C. Porras, A. J. Fernández-Leiva, Competitive algorithms for coevolving both game content and ai. a case study: Planet wars, IEEE Transactions on Computational Intelligence and AI in Games 8 (2015) 325–337.

[31] L. H. Lelis, W. M. Reis, Y. Gal, Procedural generation of game maps with human-in-the-loop algorithms, IEEE Transactions on Games 10 (2017) 271–280.

[32] D. Oranchak, Evolutionary algorithm for generation of entertaining shinro logic puzzles, in: European Conference on the Applications of Evolutionary Computation, Springer, 2010, pp. 181–190.

[33] D. Ashlock, Automatic generation of game elements via evolution, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, IEEE, 2010, pp. 289–296.

[34] A. M. Smith, M. Mateas, Answer set programming for procedural content generation: A design space approach, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 187–200.

[35] D. A. Ashlock, S. P. Gent, K. M. Bryden, Evolution of l-systems for compact virtual landscape generation, in: 2005 IEEE Congress on Evolutionary Computation, volume 3, IEEE, 2005, pp. 2760–2767.

[36] M. Frade, F. F. de Vega, C. Cotta, Evolution of artificial terrains for video games based on accessibility, in: European Conference on the Applications of Evolutionary Computation, Springer, 2010, pp. 90–99.

[37] J. Togelius, M. Preuss, G. N. Yannakakis, Towards multiobjective procedural map generation, in: Proceedings of the 2010 workshop on procedural content generation in games, ACM, 2010, p. 3.

[38] D. Loiacono, L. Cardamone, P. L. Lanzi, Automatic track generation for high-end racing games using evolutionary computation, IEEE Transactions on computational intelligence and AI in games 3 (2011) 245–259.

[39] D. Ashlock, C. Lee, C. McGuinness, Search-based procedural generation of maze-like levels, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 260–273.

[40] E. J. Hastings, R. K. Guha, K. O. Stanley, Evolving content in the galactic arms race video game, in: 2009 IEEE Symposium on Computational Intelligence and Games, IEEE, 2009, pp. 241–248.

[41] A. Martin, A. Lim, S. Colton, C. Browne, Evolving 3d buildings for the prototype video game subversion, in: European Conference on the Applications of Evolutionary Computation, Springer, 2010, pp. 111–120.

[42] E. J. Hastings, K. O. Stanley, Interactive genetic engineering of evolved video game content, in: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, ACM, 2010, p. 8.

[43] T. Tutenel, R. M. Smelik, R. Lopes, K. J. De Kraker, R. Bidarra, Generating consistent buildings: a semantic approach for integrating procedural techniques, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 274–288.

[44] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, K. O. Stanley, Petalz: Search-based procedural content generation for the casual gamer, IEEE Transactions on Computational Intelligence and AI in Games 8 (2015) 244–255.

[45] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 172–186.

[46] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (pcgml), IEEE Transactions on Games 10 (2018) 257–270.

[47] S. Kent, Model driven engineering, in: M. Butler, L. Petre, K. Sere (Eds.), Integrated Formal Methods, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 286–298.

[48] Meta Object Facility, Meta object facility (MOF) version 2.4.1, 2013. Object Management Group (OMG) Specification.

[49] J. Font, L. Arcega, Ø. Haugen, C. Cetina, Feature location in models through a genetic algorithm driven by information retrieval techniques, in: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, MODELS '16, ACM, New York, NY, USA, 2016, pp. 272–282. URL: http://doi.acm.org/10.1145/2976767.2976789. doi:10.1145/2976767.2976789.

[50] L. Arcega, J. Font, C. Cetina, Evolutionary algorithm for bug localization in the reconfigurations of models at runtime, in: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018, 2018, pp. 90–100. URL: https://doi.org/10.1145/3239372.3239392. doi:10.1145/3239372.3239392.

[51] L. Arcega, J. Font, Ø. Haugen, C. Cetina, An approach for bug localization in models using two levels: model and metamodel, Software & Systems Modeling (2019).

[52] V. R. Basili, The role of experimentation in software engineering: Past, current, and future, Berlin, Germany, 1996, pp. 442–449.

[53] G. H. Travassos, M. de Oliveira Barros, Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering, in: Proceedings of the ESEIW 2003 Workshop on Empirical Studies in Software Engineering (WSESE '03), IEEE Computer Society, Roman Castles, Italy, 2003.

[54] F. Shull, J. Singer, D. I. K. Sjberg, Guide to Advanced Empirical Software Engineering, 1st ed., Springer Publishing Company, Incorporated, 2010.

[55] L. Arcega, J. Font, Ø. Haugen, C. Cetina, On the influence of models at run-time traces in dynamic feature location, in: A. Anjorin, H. Espinoza (Eds.), Modelling Foundations and Applications, Springer International Publishing, Cham, 2017, pp. 90–105.

[56] V. R. Basili, G. Caldiera, H. D. Rombach, The goal question metric approach, in: Encyclopedia of Software Engineering, John Wiley & Sons, 1994.

[57] C. Browne, F. Maire, Evolutionary game design, IEEE Trans. Comput. Intellig. and AI in Games 2 (2010) 1–16.

[58] D. S. Johnson, A theoreticians guide to the experimental analysis of algorithms, Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges 59 (2002) 215–250.

[59] A. Arcuri, G. Fraser, Parameter tuning or default values? an empirical investigation in search-based software engineering, Empirical Software Engineering 18 (2013) 594–623.

[60] J. M. Thompson, Defining the abstract, The Games Journal (2000).

[61] I. Althöfer, Computer-aided game inventing, Technical Report, Friedrich Schiller Universität Jena (2003).

[62] H. Iida, K. Takahara, J. Nagashima, Y. Kajihara, T. Hashimoto, An application of game-refinement theory to mah jong, in: Entertainment Computing - ICEC 2004, Third International Conference, Eindhoven, The Netherlands, September 1-3, 2004, Proceedings, 2004, pp. 333–338. URL: https://doi.org/10.1007/978-3-540-28643-1{_}41. doi:10.1007/978-3-540-28643-1_41.

[63] C. Browne, Connection Games: Variations on a Theme, AK Peters, Natick, Massachussetts, 2005.

[64] W. Kramer, What makes a game good?, The Games Journal (2000).

[65] M. Gardner, Theory of everything, The New Criterion 23 (2004).

[66] H. Ellis, Impressions and Comments, Houghton Mifflin, Boston, Massachussetts, 1914.

[67] G. D. Birkhoff, Aesthetic Measure, Harvard University Press, Cambridge, Massachussetts, 1933.

[68] G. Stiny, J. Gips, Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts, University of California Press, Berkeley, California, 1978.

[69] A. Arcuri, L. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Softw. Test. Verif. Reliab. 24 (2014) 219–250.

[70] S. Garca, A. Fernndez, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044 – 2064. Special Issue on Intelligent Distributed Information Systems.

[71] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Inf. Sci. 180 (2010) 2044–2064.

[72] R. J. Grissom, J. J. Kim, "Effect sizes for research: A broad practical approach, Mahwah, NJ: Earlbaum, 2005.

[73] A. Vargha, H. D. Delaney, A critique and improvement of the cl common language effect size statistics of mcgraw and wong, Journal of Educational and Behavioral Statistics 25 (2000) 101–132.

[74] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, J. R. Woodward, Genetic improvement of software: A comprehensive survey, IEEE Transactions on Evolutionary Computation 22 (2018) 415–432.

[75] D. E. Goldberg, Computer-aided gas pipeline operation using genetic algorithms, Ph.D. dissertation (1983).

[76] M. De Oliveira Barros, A. C. Dias-Neto, 0006/2011-threats to validity in search-based software engineering empirical studies, RelaTeDIA 5 (2011).