TOWARDS SEARCH-BASED GAME SOFTWARE ENGINEERING

Daniel **Blasco Latorre**

FEBRUARY 2024

Thesis Supervisors
Dr. Carlos **Cetina Englada**
Dr. Óscar **Pastor López**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Towards Search-based Game Software Engineering

February 2024

Author:	Daniel Blasco Latorre
Thesis Supervisors:	Dr. Carlos Cetina Englada
	Dr. Óscar Pastor López

Dedication

To Alba. You make it all make sense.

To Ehren, for every second that you give me.

To Pilar, Carmen and Manuel, for you showed me the kind of person that I aspire to be.

As for you, Okami: Thanks for listening.

Acknowledgments

I want to express my deep gratitude to my directors, Dr. Carlos Cetina and Dr. Óscar Pastor, for their wisdom and guidance. They believed in this work from the very beginning, and their invaluable help made it turn into a reality.

I also want to thank the SVIT Research Group at Universidad San Jorge, an amazing group of people that perfectly embodies what a team should be. Thank you for the immense generosity, help, patience and warmth that you brought to this journey.

Last, but not least, thank you to my loved ones for always supporting me and being the compass that reminds me of what really matters.

Abstract

Video games are multidisciplinary projects which involve software development to a significant extent. This thesis tackles the software aspect of video game development through Search-based Engineering. Specifically, the objective of this work is to leverage the characteristics of video games towards Search-based Game Software Engineering, including the use of video game simulations to guide the search, a fine-grained encoding, and improvement genetic operations.

The approaches proposed outperform the baselines in maintenance (requirement traceability) and content creation (NPC generation) tasks. Maintenance and content creation are often essential tasks to ensure player retention by means of updates or expansions. In addition, this research addresses the need for industrial case studies.

This thesis presents a compendium that includes three papers produced through the research and published in academic journals, with results that show that Search-based Game Software Engineering approaches can provide improved solutions, in terms of quality and time cost.

Resumen

Los videojuegos son proyectos multidisciplinares que implican, en buena medida, el desarrollo de software. Esta tesis trata la faceta del desarrollo de videojuegos relativa al software mediante la Ingeniería del Software basada en Búsqueda (SBSE, Search-based Software Engineering). El objetivo específico de este trabajo es valerse de las características de los videojuegos en pro de una Ingeniería del Software de Videojuegos basada en Búsqueda (SBGSE, Search-based Game Software Engineering), incluyendo el uso de simulaciones de videojuegos para guiar búsquedas, codificación de granularidad fina y operaciones genéticas de mejora.

Las aproximaciones propuestas superan a las de referencia en mantenimiento (trazabilidad de requisitos) y creación de contenido (generación de NPCs). El mantenimiento y la creación de contenido son, a menudo, tareas esenciales para garantizar la retención de usuarios por medio de actualizaciones o expansiones. Además, esta investigación aborda la necesidad de estudios de caso industriales.

Esta tesis presenta un compendio que incluye tres artículos realizados durante el proceso de investigación y publicados en revistas académicas, con resultados que muestran que las aproximaciones de la Ingeniería del Software de Videojuegos basada en Búsqueda (SBGSE, Search-based Game Software Engineering) pueden mejorar la calidad de las soluciones generadas, así como reducir el tiempo necesario para producirlas.

Resum

Els videojocs són projectes multidisciplinaris que impliquen, en bona part, el desenvolupament de software. Aquesta tesi tracta la faceta del desenvolupament de videojocs relativa al software mitjançant l'Enginyeria del Software basada en Cerca (SBSE, Search-based Software Engineering). L'objectiu específic d'aquest treball és valdre's de les característiques dels videojocs en pro d'una Enginyeria del Software de Videojocs basada en Cerca (SBGSE, Search-based Game Software Engineering), incloent-hi l'ús de simulacions de videojocs per a guiar cerques, codificació de granularitat fina i operacions genètiques de millora.

Les aproximacions proposades superen a les de referència en manteniment (traçabilitat de requisits) i creació de contingut (generació de NPCs). El manteniment i la creació de contingut són, sovint, tasques essencials per a garantir la retenció d'usuaris per mitjà d'actualitzacions o expansions. A més, aquesta investigació aborda la necessitat d'estudis de cas industrials.

Aquesta tesi presenta un compendi que inclou tres articles realitzats durant el procés d'investigació i publicats en revistes acadèmiques, amb resultats que mostren que les aproximacions de l'Enginyeria del Software de Videojocs basada en Cerca (SBGSE, Search-based Game Software Engineering) poden millorar la qualitat de les solucions generades, així com reduir el temps necessari per a produir-les.

Contents

Dedication iii

Acknowledgments v

Abstract - Resumen - Resum vii

Contents xiii

I Introduction 1

II Compendium of Articles 25

1 A Fine-Grained Requirement Traceability Evolutionary Algorithm: Kromaia, a Commercial Video Game Case Study **27**

2 An Evolutionary Approach for Generating Software Models: The case of Kromaia in Game Software Engineering **71**

3 Procedural Content Improvement of Game Bosses with an Evolutionary Algorithm 119

III Discussion 173

IV Conclusions 183

Part I

Introduction

Introduction

This part introduces the context and objectives of this thesis. In addition, it includes a development overview of the research process, and describes the methodology applied. The last sections of the chapter present the structure of this work and the scientific articles that are included in the document.

Context

This thesis addresses the software aspect of video game development through Search-based Engineering. Consequently, this compendium document starts with contextual introductions to Search-based Engineering within Software Engineering, and video game research, respectively.

Software Engineering

The origins of Software Engineering can be traced back to the 20th century when, after the end of World War II, the knowledge acquired during that period led to the creation of entities like the Association for Computing Machinery (ACM) [12] and the National Aeronautics and Space Administration (NASA) [27]. Scientists from these and other institutions, like Margaret Hamilton, Douglas T. Ross, and Anthony Oettinger coined the term "Software Engineering" (SE) [17, 25]. Since then, many fields have emerged within SE: Some of them originated as cross-disciplinary areas, like Software Architecture [15], Software Testing [21], Model-driven Engineering [14], or Software Product Lines [22]; the development of other fields was motivated by the rise of web and mobile technologies, or the relevance of the Internet of Things [9]; more recently, areas like Artificial Intelligence [18, 7, 20], Green Computing [16], or Search-based Software Engineering [11] —to name a few, since the example list is not intended to be exhaustive— are trending Software Engineering topics.

Search-based Software Engineering

In 2001, Mark Harman and Bryan F. Jones introduced the term Search-based Software Engineering in a work that made SBSE be a recognized and established field of study [11], even if the problems addressed by SBSE were present in past works [10]. SBSE focuses on the application of search-based algorithms to Software Engineering contexts in order to produce near optimal solutions. SBSE involves the reformulation of Software Engineering problems as search problems by defining, for each problem studied, the following ingredients:

- A representation or encoding of the solution candidates that contains the information required to make possible the study of such candidates in order to rank, combine or alter them.
- A fitness function or criterion. This function represents how good a solution candidate is in comparison to others, taking into consideration the information provided by the encoding.
- A set of operators that allow, for instance, the creation of new solution candidates by mixing the encoded information of others that already exist, or the mutation of a solution candidate by introducing changes in its encoded representation.

SBSE has been applied successfully to a wide range of general engineering problems [11] and Software Engineering fields of study, like Model-Driven Engineering [5]. SBSE is specially useful in the context of problems for which the production of perfect solutions is difficult, making the search for near optimal solutions more practical, once the appropriate tolerance thresholds and quality criteria for a specific problem are defined.

Video Game Development

Commercial video games are complex products with specific needs and facets which set those projects apart from other software works. Video game development includes, among other aspects: Game Design, referred to the design of the abstract game content managed, rules, and mechanics that are part of the final user experience, in the same manner as traditional, non-digital games; Art Production, which involves the creation and management of any visual or sound artistic element that is included in the game; Game Software, which coordinates and integrates all the different elements of the production of the video game into the final application that will be run by users. The creation and use of such software is involved in many different ways in the development of commercial video games, including:

- External utilities like libraries, and tools dealing with tasks like parsing or conversion.

- Source code specifically written by the developers in order to realize a set of requirements, in terms of game mechanics and user experience. In fact and, in addition to source code, it is possible to use models in order to develop video game content.

- Game Engines, which are frameworks consisting of varied reusable components and tools. They boost the development by automatizing some parts of the process and assisting the developers with the expansion of the game architecture source code and the management of elements like 3D meshes, textures, sound clips, among other production assets. The use of engines means the implicit inclusion in the final application of source code that is not directly added by the developers of the game. Game Engines can be proprietary technology produced by the game developers themselves, but in the recent years the vast majority of companies use external and commercial Game Engines.

- New content that is added to an already existing game. These content items, like new characters or levels/stages, may involve new source code specifically written for them or external files which are interpreted and translated into its source code realization at runtime by a Game Engine. The creation of video game content that involves source code can be done directly writing such code, or by means of software models supported by the engine used. While code lets developers control the characteristics of the content with more detail, the level of abstraction of software models is closer to the video game domain and further from language or implementation related constraints, allowing for a more content-focused development, as shown by Figure 1. Commercial engines like Unity [24] or Unreal [8] provide modelling languages which, along with UML, are shown by Model-Driven game development literature to be used by developers [29].

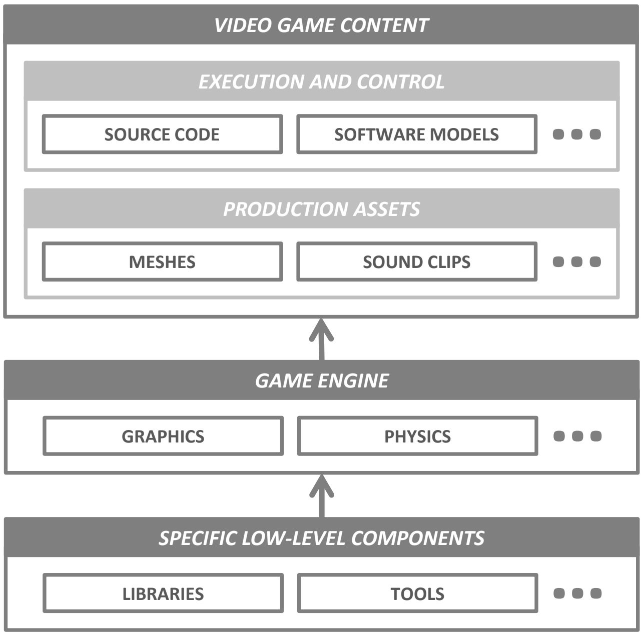


Figure 1: Video game development abstraction level outline.

Game Software Engineering

Game Software Engineering (GSE) consists in the application of Software Engineering techniques and methodologies to the specific context of video game development. In the beginning of the 21st century, GSE was compared for the first time with classic Software Engineering [19], and years later literature review works showed a growing interest of researchers in that area [1]. Such surveys evinced the following issues with regard to research diversity:

- The low number of GSE works dealing with case studies.
- A significantly low number of GSE works that include experiments in comparison with traditional Software Engineering researches.

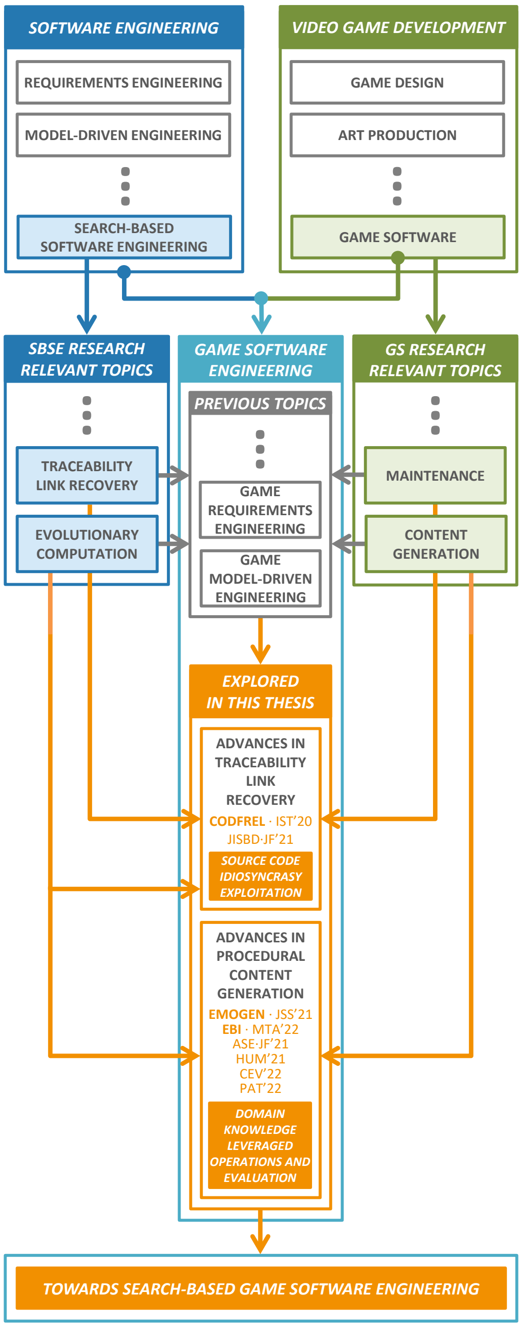

SOFTWARE ENGINEERING
REQUIREMENTS ENGINEERING
MODEL-DRIVEN ENGINEERING
SEARCH-BASED SOFTWARE ENGINEERING
VIDEO GAME DEVELOPMENT
GAME DESIGN
ART PRODUCTION
GAME SOFTWARE
SBSE RESEARCH RELEVANT TOPICS
TRACEABILITY LINK RECOVERY
EVOLUTIONARY COMPUTATION
GAME SOFTWARE ENGINEERING
PREVIOUS TOPICS
GAME REQUIREMENTS ENGINEERING
GAME MODEL-DRIVEN ENGINEERING
GS RESEARCH RELEVANT TOPICS
MAINTENANCE
CONTENT GENERATION
EXPLORED IN THIS THESIS
ADVANCES IN TRACEABILITY LINK RECOVERY
CODFREL · IST'20
JISBD·JF'21
SOURCE CODE IDIOSYNCRASY EXPLOITATION
ADVANCES IN PROCEDURAL CONTENT GENERATION
EMOGEN · JSS'21
EBI · MTA'22
ASE·JF'21
HUM'21
CEV'22
PAT'22
DOMAIN KNOWLEDGE LEVERAGED OPERATIONS AND EVALUATION
TOWARDS SEARCH-BASED GAME SOFTWARE ENGINEERING

Figure 2: Thesis context and research overview.

Objectives and Development

Goals Pursued

Given the background described, the objective of this work is to improve GSE, exploring the use of SBSE approaches implemented in such a way that they leverage the characteristics of video games.

In the articles included in this work, the term *approach* refers to the development of an artifact that interacts with a context represented by a case study. The performance of the approaches implemented in this thesis was compared with baselines in order to measure the value of the results produced.

Specifically, this thesis aims to answer the following research questions:

RQ$_1$: *How can specific video game domain characteristics be used in order to develop SBSE approaches that address GSE problems?*

RQ$_2$: *In case that video game focused SBSE approaches are successfully developed, what advances or improvements do they bring to GSE?*

Industrial Case Study Involvement

The research associated to the articles included in this thesis produced a set of approaches: CODFREL (Code Fragment-based Requirement Location), [2], EMoGen (Evolutionary Model Generation) [3], and EBI (Evolutionary Boss Improvement) [4]. These approaches have been evaluated in the context of an industrial case study: Kromaia, a commercial video game released for PC (Steam) and PlayStation 4 platforms in digital and physical formats. The following summary describes the main characteristics of that product as a commercial video game case study, in the context of the articles presented:

- Apart from external libraries, Kromaia includes over 260,000 lines of private source code written in C++ that define both a versatile game engine and specific code for the title. Besides, the architecture used is a complex blend of inheritance and composition that is aligned with those of other professional engines, like Unity [24].

- It was possible to access the Version Control System used during the development of Kromaia and, therefore, to know data such as development time estimations for game content items. In addition, the developers provided requirement specifications and the corresponding source code realizations.
- The developers of Kromaia created a Domain-Specific Modelling Language which allows for the definition of game setups, worlds, stages, characters, or contraptions as models. The game includes a model interpreter that transforms those models, which are stored in external files, into the equivalent C++ run-time objects.

Research Development Progress

The top part of Figure 2 shows how the starting point of this work was the study of the research contexts of Search-based Software Engineering and Game Software within Software Engineering and Video Game Development, respectively. The center part of Figure 2 shows the research works that are included in this thesis and deal with Requirement Traceability, Procedural Content Generation, and Procedural Content Improvement, respectively. Next, those works are introduced.

`Requirement Traceability`

Systematic Literature Review works [1] described that the most relevant research topic in the GSE community was the study of Game Requirements, which is aligned with the aim of TLR in relation with requirements. Consequently, the first work included in this thesis focused on Requirement Traceability in GSE research, incorporating aspects that take advantage of certain particularities of Game Software [2].

In addition, the fact that the evaluation was performed in a commercial video game case study introduced factors that had to be considered in the design of the approach developed for this work. The highly dispersed distribution of requirement realization code lines within the source code of a commercial video game, which is a huge solution space to search, led to the use of one of the main paradigms within SBSE: Evolutionary Computation [11]. The particularities of Game Software also led the approach to be a novel fine-grained approach in order to work with code fragments as potential requirement realizations (instead of dealing with complete methods as the state of the art).

Additionally, the approach developed made use of the knowledge that the game developers of the video game case study had in relation to the higher importance of certain terms (denoted as keywords in the work) in a requirement description. Therefore, the SBSE aspects brought to GSE and presented in this work include:

- A fine-grained management of the potential solutions proposed as requirement realizations. This is related with the architecture peculiarities found in most video games, due to the real-time control of the entities updated and their relationships, as shown by modern commercial game engines [23]. This influences the realizations of the requirements within the game source code, which tend to be highly dispersed, involving code lines from different and distant methods.
- The use of the game developer's domain knowledge, in order to identify important terms in requirements that are specified by means of natural language. The keywords identified were used in the genetic operations in order to help with the guidance of the evolutionary algorithm.

The results showed that the approach, developed with a fine-grained design and developer knowledge-powered genetic operations, outperformed the results obtained with a widely used baseline. This work was published as an article (labelled as IST'20 in this document) in the Information and Software Technology journal. It was also presented as a journal-first paper in the Spanish Conference on Software Engineering and Databases (JISBD 2021), in the context of the Computer Science Spanish Conference (CEDI 20/21). The contribution to this thesis is represented in the bottom center part of Figure 2.

The results obtained in the first work suggested that the importance of tacit knowledge prevented the solutions produced from being optimal, especially in cases like requirement descriptions that are defined with natural language and often lack domain knowledge that remains unwritten or undocumented.

Procedural Content Generation

The next work tackles Procedural Content Generation (PCG), which is a hot topic in the video game research community. PCG is concerned with the algorithm-driven automated generation of game content [26]. PCG achieved success in the industry, but such success is limited to aspects like foliage generation (e.g: SpeedTree [13], used in Unity [24] and Unreal [8]).

This work shows that there is a key aspect present in video games that can boost Search-based Game Software Engineering (SBGSE) approaches: the possibility of using simulations. A game simulation is a representation of a game session that reproduces, with a variable level of accuracy and abstraction, the behaviours, interactions, and game status changes that involve all the entities that take part in the video game simulated, like players, characters, stages, or artifacts.

In terms of gameplay quality satisfaction, the interaction of a human player with a game is the best fitness method, but it is difficult to apply in the context of a search-based artifact: for instance, in the case of evolutionary algorithms, every individual produced must be measured by means of a fitness function, which prevents an unattended implementation, demands a high and arguably exhausting amount of time from human players, and substantially delays the production of results. Therefore, simulations can act as the main alternative to sessions with human players. Regardless of its level of abstraction, a simulation produces results equivalent, in terms of measurement, to those produced in an actual game session played by a human user.

In traditional software, the design and creation of simulations is often difficult to the point of requiring an implementation that is more complex than the software simulated. However, in the case of video games, simulations can be more naturally developed. This is due to the inclusion of entities like Non-Playable Characters or NPCs that act as enemies in shooters, rival pilots in racing games, or wild creatures in open world adventures.

NPCs provide a succession of events and interactions that can shape an emergent "plot" in the same way that it happens during a real game session. Due to their potential benefits and, since they are more attainable in the context of video games, simulations could play a cornerstone role in search-based PCG.

The aspects brought by this work to SBGSE include:

- The use, in the fitness function that guides the evolutionary algorithm of the approach, of game simulations in order to assess the solution candidates for game bosses, a complex type of content neglected in past works that used simulations. In this work, the simulations take into account aspects like game design and mechanics that represent complex domain knowledge. Since the models produced by the approach are meant to be loaded by the interpreter, the approach developed ultimately leverages the model interpreter.
- An encoding for the game boss models managed that allows for the reparation of encoded individuals that are not considered valid by the model interpreter of the video game case study. The reparation actions available are included in the genetic operations.
- The utilization of objective quality metrics present in the Game Research literature in order to measure the value of the solution candidates produced [6]. These measurements allowed for the comparison between the game bosses created by the human developers and those proposed as solution candidates by the approach presented.

The results obtained in the second work showed that the approach presented produced game content that was comparable to the content created manually by the human developers of the video game case study. In addition, the time required was significantly reduced. The work was published as an article (labelled as JSS'21 in this document) in the Journal of Systems and Software and was presented in the IEEE/ACM International Conference on Automated Software Engineering (ASE 2021) as a journal-first paper. It was also presented in the First Video Game Science Spanish Conference (CEV'22), created by the Video Game Science Spanish Society (SECiVi). Furthermore, a patent based on the research (labelled as PAT'22 in this document) was requested in the United States. Additionally, and in the Genetic And Evolutionary Computation Conference (GECCO 2021), the work was awarded with the Bronze Medal in the Annual "Humies" Awards for Human-Competitive Results Produced by Genetic and Evolutionary Computation (labelled as HUM'21 in this document). This was the first time that a spanish research team won an award in the 18-year history of the competition. The contributions of the work to this thesis is represented in the bottom center part of Figure 2.

Procedural Content Improvement

The next step in the research of the PCG topic was exploring the production of game bosses by means of improvement. Improvement deals with the possibility of letting developers focus on certain steps in the content creation process and delegate others to automated approaches that perform such improvement to obtain, in the end, complete content.

The developers of the video game case study created bosses through a process with clear steps that involved Creative Design, Spatial Organization, Behaviour Specification and Equipment Configuration. The last step (Equipment Configuration) deals with weaponry and weak point addition and distribution in a boss, and the bosses included originally in the game required a testing period of around a month to reach a proper configuration that could be considered acceptable and satisfactory by the players.

This work intended to automate that last step in order to produce boss candidates with good quality and in less time. In addition to the use of an evolutionary algorithm, a simulation-based fitness and widely used quality metrics for measuring the results, the aspects presented in this work include:

- The use of partially generated bosses as input, which are encoded and fed to an evolutionary algorithm as the starting individuals of the population managed.
- Content improvement-oriented Crossover and mutation genetic operations which contribute to the genetic improvement of the individuals involved. Once two individuals produce a new one, or when an individual mutates by means of the Crossover or the mutation operations, respectively, such operations take advantage on the boss metamodel. The metamodel is used in order to determine, for instance, the elements of the boss model that should be added in order to make the boss be valid after the genetic operation. Those elements are not explicitly included in the encoding and could be related with steps that occur before Equipment Configuration, which is the stage that the approach developed focuses on. Therefore, the improvement affects the initial, partially generated content that the approach was given as input, and those implicit changes become evident when the final, complete content produced is decoded.

The results showed that the approach developed in the third work provided game content that was comparable in terms of quality with the bosses created by the developers of the original version of the video game case study. However, the time required in order to produce complete content was considerably lower. The work was published as an article (labelled as MTA'22 in this document) in the Multimedia Tools and Applications Journal, and the contribution to this thesis, as a continuation of the previous work, is represented in the bottom center section of Figure 2.

Research Methodology

The development process of this thesis was planned taking into consideration the methodology proposed by Roel J. Wieringa in Design science Methodology for Information Systems and Software Engineering [28]. The motivation behind the use of this methodology is the objective of contributing to certain Problem Contexts by designing artifacts that interact with those contexts through an empirical cycle. In Design Science, such interaction between an artifact and a Problem Context is denoted as Treatment. In [2], [3], and [4], the research outcomes include treatments that consist in the application of approaches that run unattended and are intended to improve, in terms of quality and time, the performance of baselines in the Problem Contexts of Traceability Link Recovery, and Procedural Content Generation/Improvement, respectively. The works included in the thesis used the Design Science methodology, applying the following empirical cycle, as shown in Figure 3:

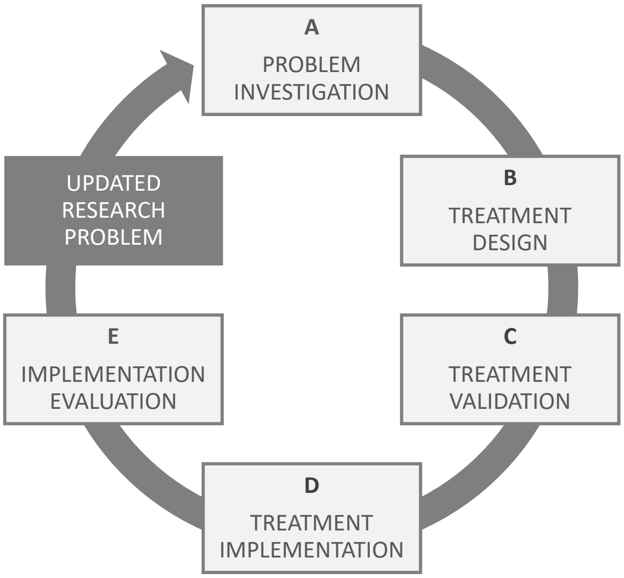


Figure 3: Empirical cycle used in the Design Science methodology.

A Problem Investigation: In this stage, previous research works were studied in order to elucidate how SBSE had been used in the past to address Game Software Research and the potential of the application of GSE to explore advances and contributions in Game development hot topics. After such investigation, the research questions were defined.

B Treatment Design: In the next stage, the approaches that were necessary to answer the research questions were designed, and the experimental setup (that took into account baselines selected during the Problem Investigation) was specified.

C Treatment Validation: This stage implied validating the treatment designed by considering the threats to validity that were detected in the Treatment Design stage. In order to minimize the possible impact of those threats, various measures were adopted, like applying widely accepted indicators, or evaluating the approaches in industrial case studies.

D Treatment Implementation: This stage included both the implementation of the approaches required and the specification of the experiments that allowed the interaction of those approaches with the Problem Context

E Implementation evaluation: In this stage, the execution of the experiments that involved the treatments took place. In addition, those experiments produced results, that were used in order to analyze and discuss the outcomes of the research with regard to the research questions.

The last stage, which studies the degree of success of the treatment, determined an eventual start of a new cycle with an improved perspective on the Problem Context, and could result in a updated set of research questions or the re-design of the treatment.

Structure of the Thesis

This work is structured in accordance to the regulations in Universitat Politècnica de València with regard to PhD thesis development. More specifically, this thesis presents a compendium of articles and it is organized as follows:

I. Introduction: The first part of the thesis describes the context in which the research takes place, the objectives of this work, the methodology applied through the research, and a summary of the articles that make up the compendium.

II. Compendium of Articles: The second part includes the collection of articles written during the development of the research. These articles were all published in specialized journals and they are presented following the format of the thesis, while keeping the original content and bibliographies.

III. Discussion: The third part discusses the results obtained through the research, taking into account the different scopes covered by this thesis.

IV. Conclusions: The last part of the thesis presents the conclusions of this work, given the objectives that were set and the outcomes of the research process.

Compendium Description

The following articles have been produced and published in different journals in the course of the research process presented in this thesis:

1. **A Fine-grained Requirement Traceability Evolutionary Algorithm: Kromaia, a Commercial Video Game Case Study** (Information and Software Technology) (Blasco, Cetina, Pastor, 2020) (**IST'20**) [2].
   This work addresses the problem of Traceability Link Recovery in commercial video games, taking into account the size of the source code in industrial products and the inherent requirement dispersion of real time simulation game architectures. The CODFREL approach proposed in this article advances in this field by making use of game developers' knowledge and a fine-grained-based search in order to improve the accuracy of the solutions that are provided as requirement realization candidates.

2. **An Evolutionary Approach for Generating Software Models: The Case of Kromaia in Game Software Engineering** (The Journal of Systems and Software) (Blasco, Font, Zamorano, Cetina, 2021) (**JSS'21**) [3].
   This research focuses on Procedural Content Generation in video

games (more specifically, game bosses) for commercial products, from a Game Software Engineering point of view. This work led to the inclusion, with evolutionary algorithm guidance purposes, of a fitness function based on game simulations that are designed and calibrated after the domain specific knowledge provided by game developers. In addition, the work includes the use of video game-specific objective quality measures in order to compare the game content produced by the EMoGen approach presented with the content created by human developers.

3. **Procedural Content Improvement of Game Bosses with an Evolutionary Algorithm** (Multimedia Tools and Applications) (Blasco, Font, Pérez, Cetina, 2022) (**MTA'22**) [4].
   This article continues the research on the procedural generation of content for commercial video games, considering the case of the need for complete content that must be produced taking incomplete and partially generated content as starting points.

Additionally, two articles included in the compendium were presented in relevant national and international conferences as journal-first papers as follows:

- IST'20 was presented in the Spanish Conference on Software Engineering and Databases (JISBD 2021), in the context of the Computer Science Spanish Conference (CEDI 20/21).
- JSS'21 was presented in the IEEE/ACM International Conference on Automated Software Engineering (ASE 2021).

In 2021, during the Genetic and Evolutionary Computation Conference (GECCO 2021), JSS'21 was awarded with the Bronze Medal in the 18th Annual "Humies" Awards for Human-Competitive Results Produced by Genetic and Evolutionary Computation (HUM'21). Besides, JSS'21 was presented in 2022 in the First Video Game Spanish Conference (CEV'22), organized by the Video Game Science Spanish Society (SECiVi).

In 2022, the application corresponding the patent "Method and System for Automatic Synthesis of Videogame Assets" (PAT'22) by Jaime Font, Daniel Blasco, and Carlos Cetina, based in the methods and techniques applied in JSS'21 and MTA'22, was ready for examination in the United States Patent and Trademark Office.

Research Support

The research works included as a compendium in this thesis have been partially funded as follows:

- [2] and [3] have been supported in part 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).
- [4] 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 VARIATIVA under Grant PID2021-128695OB-I00, and in part by the Gobierno de Aragon (Spain) (Research Group S05 20D).

Bibliography

[1] Apostolos Ampatzoglou and Ioannis Stamelos. "Software engineering research for computer games: A systematic review". In: *Information and Software Technology* 52.9 (2010), pp. 888–901. ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2010.05.004` (cit. on pp. 7, 10).

[2] Daniel Blasco, Carlos Cetina, and Oscar Pastor. "A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study". In: *Information and Software Technology* 119 (2020). ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2019.106235` (cit. on pp. 9, 10, 16, 18, 20).

[3] Daniel Blasco et al. "An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering". In: *Journal of Systems and Software* 171 (2021), p. 110804. ISSN: 0164-1212. DOI: `https://doi.org/10.1016/j.jss.2020.110804` (cit. on pp. 9, 16, 18, 20).

[4] Daniel Blasco et al. "Procedural content improvement of game bosses with an evolutionary algorithm". In: *Multimedia Tools and Applications* 82 (2022), pp. 1–33. ISSN: 1573-7721. DOI: `https://doi.org/10.1007/s11042-022-13674-6` (cit. on pp. 9, 16, 19, 20).

[5] Ilhem Boussaïd, Patrick Siarry, and Mohamed Ahmed-Nacer. "A survey on search-based model-driven engineering". In: *Automated Software Engineering* 24.2 (2017), pp. 233–294 (cit. on p. 5).

[6] Cameron Browne and Frédéric Maire. "Evolutionary Game Design". In: *IEEE Trans. Comput. Intellig. and AI in Games* 2.1 (2010), pp. 1–16. DOI: `10.1109/TCIAIG.2010.2041928` (cit. on p. 13).

[7] Bruce G Buchanan. "A (very) brief history of artificial intelligence". In: *Ai Magazine* 26.4 (2005), pp. 53–53 (cit. on p. 4).

[8] Epic Games. *Unreal Engine, Version 2018.3.9.* Cary, North Carolina, 1998 (cit. on pp. 6, 12).

[9] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. "The internet of things". In: *Scientific American* 291.4 (2004), pp. 76–81 (cit. on p. 4).

[10] M. Harman, Y. Jia, and Y. Zhang. "Achievements, Open Problems and Challenges for Search Based Software Testing". In: *IEEE 8th International Conference on Software Testing, Verification and Validation (ICST).* 2015, pp. 1–12. DOI: `10.1109/ICST.2015.7102580` (cit. on p. 4).

[11] M. Harman and B.F. Jones. "Search-Based Software Engineering". In: *Information Software Technology* 43 (2001), pp. 833–839 (cit. on pp. 4, 5, 11).

[12] Harvard University Harvard Computation Laboratory. *Notice on Organization of an Eastern Association for Computing Machinery.* 1947 (cit. on p. 4).

[13] Inc. Interactive Data Visualization. *SpeedTree.* Lexington, South Carolina, 2002 (cit. on p. 12).

[14] Stuart Kent. “Model Driven Engineering”. In: *Integrated Formal Methods.* Ed. by Michael Butler, Luigia Petre, and Kaisa Sere. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 286–298. ISBN: 978-3-540-47884-3 (cit. on p. 4).

[15] P. Kruchten, H. Obbink, and J. Stafford. “The Past, Present, and Future for Software Architecture”. In: *IEEE Software* 23.2 (2006), pp. 22–30. DOI: `10.1109/MS.2006.59` (cit. on p. 4).

[16] Patrick Kurp. “Green computing”. In: *Communications of the ACM* 51.10 (2008), pp. 11–13. DOI: `10.1145/1400181.1400186` (cit. on p. 4).

[17] Michael S. Mahoney. “The Roots of Software Engineering”. In: *CWI quarterly* 3 (1990), pp. 325–334 (cit. on p. 4).

[18] Pamela McCorduck et al. “History of artificial intelligence.” In: *IJCAI.* 1977, pp. 951–954 (cit. on p. 4).

[19] Michael McShaffry. *Game Coding Complete.* Paraglyph Publishing, 2003. ISBN: 1932111751 (cit. on p. 7).

[20] Nikesh Muthukrishnan et al. “Brief history of artificial intelligence”. In: *Neuroimaging Clinics* 30.4 (2020), pp. 393–399 (cit. on p. 4).

[21] Jiantao Pan. “Software testing”. In: *Dependable Embedded Systems* 5.2006 (1999), p. 1 (cit. on p. 4).

[22] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 3540243720 (cit. on p. 4).

[23] Unity Technologies. *Unity - Manual: Order of execution for event functions, Version 2022.3.* San Francisco, California, 2022 (cit. on p. 11).

[24] Unity Technologies. *Unity, Version 2018.3.9.* San Francisco, California, 2005 (cit. on pp. 6, 10, 12).

[25] Matti Tedre. *The Science of Computing: Shaping a Discipline.* CRC Press, 2014. ISBN: 9781482217704 (cit. on p. 4).

[26] Julian Togelius et al. "Search-Based Procedural Content Generation: A Taxonomy and Survey." In: *IEEE Trans. Comput. Intellig. and AI in Games* 3.3 (2011), pp. 172–186 (cit. on p. 12).

[27] 85th United States Congress. *National Aeronautics and Space Act.* 1958 (cit. on p. 4).

[28] Roel J Wieringa. *Design science methodology for information systems and software engineering.* Springer, 2014 (cit. on p. 15).

[29] Meng Zhu and Alf Wang. "Model-driven Game Development: A Literature Review". In: *ACM Computing Surveys* 52.6 (2019), pp. 1–32. ISSN: 0360-0300. DOI: `10.1145/3365000` (cit. on p. 6).

Part II

Compendium of Articles

Chapter 1

A Fine-Grained Requirement Traceability Evolutionary Algorithm: Kromaia, a Commercial Video Game Case Study

Commercial video games usually feature an extensive source code and requirements that are related to code lines from multiple methods. Traceability is vital in terms of maintenance, so it is necessary to explore such search spaces properly. This work presents and evaluates CODFREL (Code Fragment-based Requirement Location), our approach to fine-grained requirement traceability, which lies in an evolutionary algorithm and includes encoding and genetic operators to manipulate code fragments that are built from source code lines. We compare it with a baseline approach (Regular-LSI) by configuring both approaches with different granularities. We evaluated our approach and Regular-LSI in the Kromaia case study. The approaches are configured with method and code line granularity and work on 20 requirements that are provided by the development company. Our approach and Regular-LSI calculate similarities between requirements and code fragments or methods to propose possible solutions and, in the case of CODFREL, to guide the evolutionary algorithm. The results show that our approach outperforms Regular-LSI in precision and recall, with values that are 26 and 8 times better, respectively, even though it does not achieve the optimal solutions.

1.1 Introduction

Traceability has been shown by researchers to have a significant impact on successful software engineering projects [42]. This has encouraged reliability and maintainability improvement efforts to trace and verify critical, non-reliable sections in software systems [24]. Traceability Link Recovery (TLR) has been studied by software engineers for a considerable number of years [25], [37], and low defect rates in software products are associated to more complete Traceability [33].

In the video game industry, the titles classified as AAA are those produced and distributed by a mid-sized or major publisher and are typically more complex than regular games and have high development and marketing budgets. Due to the nature of real-time physics simulation, high numbers of interacting entities, and long chains of events that branch and have a significant impact on the course of the game, AAA video games feature many requirements that involve more than one complete method. Current traceability approaches [10] evaluate methods in the source code of a software product as atomic units. The purpose of approaches of this type is to decide which method or method set is the best candidate for a given requirement, conceiving requirements as functionalities that are likely to be mainly defined by one or various complete methods in the source code of a software product. These approaches are not suitable for AAA video games featuring dispersed requirements.

Our approach, Source Code Fragment-based Requirement Location (CODFREL), generates possible solutions that are more flexible than complete methods. Our approach relies on an evolutionary algorithm which:

- searches for flexible solution candidates, represented by **code fragments**, i.e., sets of code lines belonging to the source code that are not necessarily contiguous or included in a single method.
- explores the search space through **genetic operations** involving mutation and fusion. The search process is guided by similitude evaluation between the terms present in a code fragment and those specified by the requirement; this similitude is measured through **Latent Semantic Indexing** (LSI) [28].

Our evaluation compares our approach to Regular-LSI. Regular-LSI uses LSI, but it does not use code fragments or an evolutionary algorithm. We compare our CODFREL approach to Regular-LSI using requirements from Kromaia, a commercial video game case study. Kromaia is a physics-based space simulation video game that was developed by Kraken Empire (www.krakenempire.com) and published by Rising Star Games (www.risingstargames.com). This is a title that has been released worldwide both digitally and physically, translated to eight languages, and ported from PC to PlayStation 4.

In order to evaluate our approach and Regular-LSI, we have configured different versions of the two approaches: our CODFREL approach was studied using code fragments and complete method granularity; and Regular-LSI was configured with three different cut-off strategies to search for the best complete method, a set containing the 10 best complete methods, and a set containing the 10 best code lines, considering each code line as a method. The results obtained show that, in comparison to Regular-LSI, our approach provides better solutions, both in terms of precision and recall, for requirements that are dispersed within the source code. Precision, recall and F-measure are information retrieval metrics that are widely used [8]. These results show that, using the different configurations mentioned, our CODFREL approach outperforms Regular-LSI in precision, recall, and F-measure, with average values of 57% and 28% for precision, around 27% for recall, and 29% and 21% for F-measure. Regular-LSI obtained average values of 4%, 0.7%, and 0.1% for precision, for the different configurations used. For recall, the values were 0.5%, 0.5%, and 9%. F-measure reached values of 0.9%, 0.6%, and 0.2%.

Both our approach and Regular-LSI output a solution that has to be manually refined to obtain all of the code that is relevant to the requirement. The solutions generated by our approach are better starting points in comparison to Regular-LSI. However, our approach does not obtain perfect solutions (every relevant code line in a requirement). Tacit knowledge, which is not written in the requirements, causes this issue. Therefore, we plan to deal with this matter through reformulations that expand the requirements with descriptions provided by domain experts.

The structure used in the paper is the following: Section 2 describes the motivation for our work. Section 3 presents an overview of our CODFREL approach. Section 4 presents the code fragment encoding. Section 5 describes the operations used in code fragment generation. Section 6 discusses how code fragment suitability is determined. Section 7 deals with the evaluation, comparing our approach to Regular-LSI, and Section 8 discusses the results obtained. Section 9 deals with future work, Section 10 describes the threats to validity, and Section 11 summarizes related works. Finally, Section 12 presents our conclusions.

1.2 Motivation

Commercial video games, like Kromaia, often behave like real-time simulation applications, which coordinate internal update processes and data structures that render information as audiovisual data. This data is meant to be coherent with internal logics, but it is not necessarily directly related to them in terms of locality.

Apart from various libraries used in the project, Kromaia features a considerably high number of private code lines (over 260,000) resulting from two main blocks, both of which are owned by the development company: a proprietary game engine, and the video game source code (VGSC). The case study focuses on the VGSC, which contains over 145,000 lines of code.

The use of an evolutionary algorithm emerges as a response to a situation in which the solution space is huge and the requirements are dispersed. Taking into account the VGSC, the current traceability approaches [10] (which feature method granularity) are required to evaluate approximately 9000 complete methods. However, code fragments (the granularity in our approach) may feature any code line in the VGSC, so the total number of possible code fragments in the VGSC is over $2^{145,000}$. Our approach, CODFREL, works with code fragments and uses an evolutionary algorithm to search such a large solution space.

1.3 Overview of the CODFREL Approach

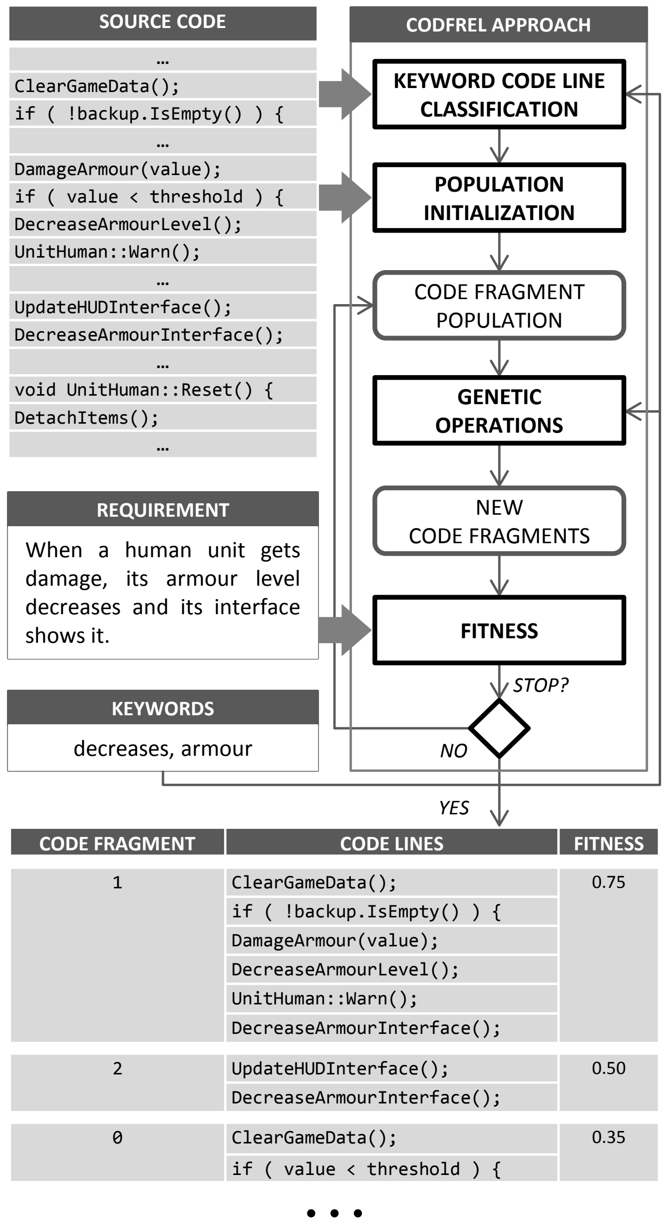
This section presents our CODFREL approach. The approach has a clear goal: to obtain the code fragment from the source code that realizes a requirement that is specified in a natural language. The evolutionary algorithm of CODFREL iterates a code fragment population, which evolves through genetic operations inspired by processes that are present in nature. This evolutionary algorithm is driven by a fitness operation that takes into consideration the requirement. In the end, the output delivered is a ranking, which is a list sorted by a fitness value that features code fragments that might realize the requirement described.

The upper left section in Fig. 1.1 shows an example of input to our CODFREL approach.

- The **Source Code**. In this paper, we evaluate our approach on the Kromaia VGSC, and we also use it as a running example.
- The **Requirement** (in natural language) to be located in our case study. The requirement is in the Game Design Document created for the video game. The requirement terms may include general vocabulary that is commonly used in the video game industry and, more specifically, terms of the specific video game that are likely to be found in the VGSC.
- In order to minimize the effects of both irrelevant and deceiving terms, our approach asks to select the most relevant terms in the requirement. These terms, the **Keywords**, help our approach to find starting points and provide guidance to the evolutionary algorithm.

The section on the right of Fig. 1.1 shows the main steps in our approach.

- First, the **Keyword Code Line Classification** step identifies those code lines from the VGSC that feature keywords, which are tagged terms in the requirement. These lines are used in both the initial code fragment population creation step and in those genetic operations involving the expansion of existing code fragments.



```
…
ClearGameData();
if ( !backup.IsEmpty() ) {
…
DamageArmour(value);
if ( value < threshold ) {
DecreaseArmourLevel();
UnitHuman::Warn();
…
UpdateHUDInterface();
DecreaseArmourInterface();
…
void UnitHuman::Reset() {
DetachItems();
…
```

When a human unit gets damage, its armour level decreases and its interface shows it.

CODE FRAGMENT	CODE LINES	FITNESS
1	`ClearGameData();`	0.75
	`if ( !backup.IsEmpty() ) {`	
	`DamageArmour(value);`	
	`DecreaseArmourLevel();`	
	`UnitHuman::Warn();`	
	`DecreaseArmourInterface();`	
2	`UpdateHUDInterface();`	0.50
	`DecreaseArmourInterface();`	
0	`ClearGameData();`	0.35
	`if ( value < threshold ) {`	

• • •

Figure 1.1: CODFREL Overview

- The **Population Initialization** step calculates the starting code fragment population, which is extracted from the VGSC using code lines determined by the keywords. Each initial code fragment is generated as follows: first, a random code line with keyword presence is selected as the starting point. Then, the code fragment is completed by adding a random number of code lines that are placed before and after the starting code line. Random initialization is a common practice in evolutionary computation.

- The **Genetic Operations** step produces a new code fragment generation. This step involves the use of a selection operator that chooses the code fragments that will be the parents of the new generation. This selection, which is done using fitness values, is meant to promote the best code fragments in the population to be parents. Then, new code fragments are produced by mixing code lines from two parents through a fusion operation. This step also introduces modifications in the new code fragments using a guided mutation operation (by adding or removing code lines from the code fragment), which hopefully would make the new code fragments reach fitness values that exceed what their parents achieved.

- The **Fitness** step evaluates the code fragments by assigning a value that depends on the similarity between each code fragment and the requirement. The code fragments that share more terms with the requirement will get the highest values.

The process is over when a code fragment features a fitness value that is greater than a predefined threshold or once a certain time limit is reached. As a result, our CODFREL approach produces a code fragment set, where each code fragment is relevant to the requirement (see the lower section in Fig. 1). The set may be organized as a ranking, ordering the code fragments by similarity to the requirement.

In summary, our CODFREL approach ultimately searches for code fragments that are relevant to the requirement. In order to succeed, the approach creates/iterates a code fragment population and searches within that population using a fitness function that evaluates code fragments by assigning values that depend on the similarity to the requirement defined in natural language.

Sections 4, 5, and 6 show how code fragments are encoded in our CODFREL approach as well as the genetic operations applied and the criteria used by the fitness function to assign different values to code fragments depending on how similar they are to the requirement.

1.4 Code Fragment Encoding of the CODFREL Approach

The code fragments generated by our approach represent the solutions proposed for the requirement. These solutions need to be encoded, a task that, in evolutionary algorithms, is usually done by storing possible solutions as arrays or strings containing binary values such as 0/1 or true/false.

In our CODFREL approach, where the solutions are the code fragments, the encoding is as follows: each code fragment is a list of code line elements; and a code line element contains indexing information that is relative to the file from which the code line was taken, its local position in that file, and its global position relative to the complete VGSC. Therefore, the encoding used in our approach to represent code fragments is not a fixed length structure, but a variable length set of code lines that is ordered, taking into account the relative positions of each of those lines in the VGSC.

1.5 Genetic Operations of the CODFREL Approach

Our CODFREL approach generates new code fragments using the existing ones as parents and making use of two genetic operators: fusion and guided mutation. We adapted these operators to function with code fragments, which represent a code line set from the VGSC.

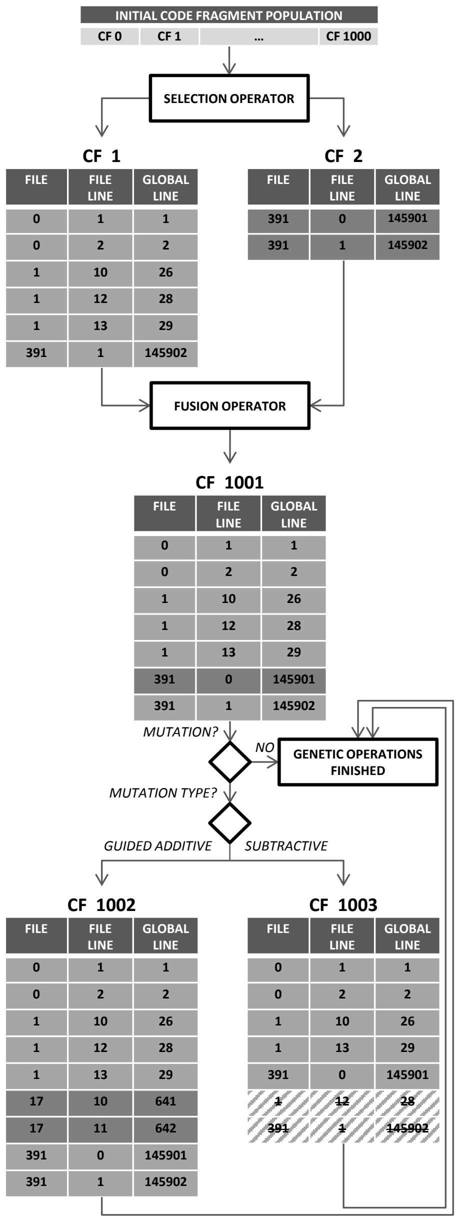
Prior to the application of the genetic operators, the best possible parents need to be selected from a given code fragment population so that the genetic operators have data to work with. For that reason, our approach uses a selection operator that is based on the widely used wheel selection method [4]: every code fragment in the population has a probability of being selected to reproduce that is proportional to its fitness value; therefore a higher fitness value implies a higher probability of being chosen to generate new code fragments.

1.5.1 Fusion

The fusion operator works like traditional evolutionary computation methods, in which a new individual is generated by combining the generic material of two existing individuals. Depending on the resulting combination and the environmental conditions, the new individual could outperform their parents or it may not even survive (or, subsequently, reproduce). In our approach, code fragments act as the reproducing individuals and the fusion operator is responsible for mixing the lines that they contain. As a result, the new code fragment contains a code line set in which the lines are ordered according to their relative position in the complete VGSC, without repetitions, in the case of lines that are present in both parents.

The fusion operator receives two code fragments and creates a new one by combining every code line in the parents, hence, without losing information. Most common recombination techniques in genetic algorithms such as crossover [14] imply inheriting parental content, partially or totally. The fusion operator could be considered as a special case of crossover that does not discard code lines from parent code fragments that have been selected as standing out and contain supposedly valuable information that is related to a requirement.

A typical case of fusion application in our CODFREL approach is shown in Fig. 1.2 with code fragment examples. The fusion operator takes two code fragments as parents (CF1 and CF2). First, the selection operator uses the wheel selection method to choose two parents. Then, the fusion operator generates a new code fragment containing the full code line sets that are present in its parents. Therefore, the new code fragment generated contains every code line in the parents, without repetitions (e.g., code line 145902 is not duplicated). The new individuals obtained



CF 1

FILE	FILE LINE	GLOBAL LINE
0	1	1
0	2	2
1	10	26
1	12	28
1	13	29
391	1	145902

CF 2

FILE	FILE LINE	GLOBAL LINE
391	0	145901
391	1	145902

CF 1001

FILE	FILE LINE	GLOBAL LINE
0	1	1
0	2	2
1	10	26
1	12	28
1	13	29
391	0	145901
391	1	145902

CF 1002

FILE	FILE LINE	GLOBAL LINE
0	1	1
0	2	2
1	10	26
1	12	28
1	13	29
17	10	641
17	11	642
391	0	145901
391	1	145902

CF 1003

FILE	FILE LINE	GLOBAL LINE
0	1	1
0	2	2
1	10	26
1	13	29
391	0	145901
~~1~~	~~12~~	~~28~~
~~391~~	~~1~~	~~145902~~

Figure 1.2: Genetic Operators: Guided Mutation and Fusion over Code Fragments

with this operator may include a higher number of code lines than their parents, as shown in Fig. 1.2.

1.5.2 Mutation

The mutation operator makes new individuals show changes in their genetic material that are caused by random factors and are not due to being inherited from their parents. The (usually) minimal variations modifying the inherited material may be positive or negative, depending on whether the mutation produces adaptive advantages or disabilities.

In our CODFREL approach, the mutation operator is applied on newly generated fragments after they have been produced by the fusion operator. However, mutations do not always occur and depend on a probability of a new code fragment being affected by a mutation. After a mutation takes place, the new code fragment may contain new lines belonging to the video game code or lose some of the lines that were featured prior to the mutation. Our approach proposes that, since the search space is huge, the creation of good starting code fragments should be guided, since it is difficult to create or improve individuals by adding random code lines from the VGSC due to its size. However, our CODFREL approach proposes that the loss of code lines is not guided, so that even the removal of lines with high similarity with a requirement could lead to code fragments that are closer to the realization of such a requirement. This proposal regarding mutations could be improved or modified in future works, taking into account the results obtained. In the end, the mutation operators provide another possible solution for the target requirement (the unmodified code fragment and its parents are also possible solutions). The nature of the mutation applied by the operator is based on random criteria:

- **Subtractive Mutation**: This type makes the mutation operator remove lines from the code fragment by randomly selecting them, as shown in the bottom right section in Fig. 1.2. The number of lines removed and the selection criteria are random, so there is no need to ensure that the lines removed are consecutive.

- **Guided Additive Mutation**: When the operator performs an additive mutation, a random number of lines belonging to the original VGSC (and not necessarily consecutive) is added to the code fragment, first selecting a starting code line that acts as a reference for the eventual code line set added. The guidance consists in having a certain probability of selecting a starting code line in the VGSC that is not completely random; instead, the operator may select a starting code line featuring terms that are tagged as keywords in the requirement (and, eventually, more code lines surrounding that line). Fig. 1.2 shows an example featuring a new code fragment, CF 1002, that was created by applying this mutation operation to CF 1001. The code lines added are 641 and 642, with 642 being the starting code line featuring a keyword:

  – `641:  if ( units[i]->IsIdle() ) {`

  – `642:  units[i]->Reset`$\boxed{\texttt{Armour}}$`Interface();`

  Code line 641, however, is a line that was added for being adjacent to 642 in the VGSC, like other lines which were not selected (but could have been) in the example.

1.6 Fitness of the CODFREL Approach

In the context of our CODFREL approach, the fitness step is intended to determine the value of each code fragment generated. Following the principles applied in evolutionary algorithms, the fitness step takes a code fragment population as input and measures the degree of adaptation of each code fragment to the environment (the adaptation being the quality of the code fragment as a solution for the requirement described). Once the step finishes, it provides a ranking in which every code is placed according to the fitness value assigned so that the top-ranked code fragments are the most similar to the requirement.

This similarity is evaluated in our approach through Latent Semantic Indexing (LSI) [28], which is currently the best performing Information Retrieval (IR) technique in terms of outcomes [34, 30, 32, 21]. In this step, it is responsible for comparing the code fragments proposed for a requirement with the requirement specified in natural language. In the context of LSI, the input for which the solution is searched is denoted as "query", while the individual elements to be evaluated as possible solutions for realizing this query are denoted as "documents". For our approach, this scheme implies that the requirement acts as a query, while code fragments are documents.

1.6.1 Natural Language Processing

Before applying LSI, the requirement is processed through well-known Natural Language Processing techniques (Part-of-Speech tagging [11] and Lemmatizing techniques [35]) so that the gap between the code fragment texts and the requirement is reduced.

- The requirement is first divided into words or **tokens**. Depending on the text complexity, separators such as spaces or semicolons may work as tokenizers (i.e., elements used to split strings), but descriptions involving in-code elements might require more sophisticated processing.
- The second stage in the requirement processing consists in removing articles, conjunctions, and other elements that do not provide useful information. This task is carried out by applying the **POS** (Part of Speech) technique. This tagging technique analyzes the grammatical role of the words in the text and helps remove the undesired material.
- Through the usage of semantic techniques such as Lemmatizing, words can be reduced to their semantic roots or **lemmas**. Thanks to lemmas, the language used in the requirement is unified, thus avoiding verb tenses, noun plurals, and strange word forms that negatively interfere with the fitness.

Fig. 1.3 shows the application of these techniques to a requirement. The token extraction step involves the use of separators. After the POS step, tokens like "a", "and", or "it" are removed for not being relevant

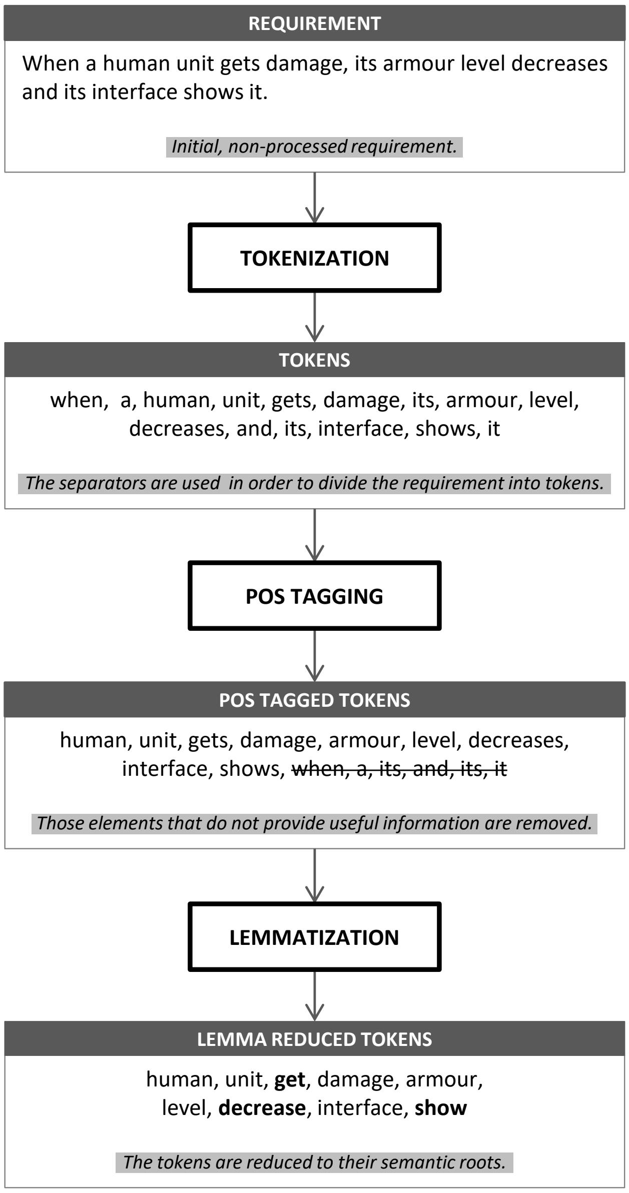

REQUIREMENT
When a human unit gets damage, its armour level decreases and its interface shows it.
Initial, non-processed requirement.
TOKENIZATION
TOKENS
when, a, human, unit, gets, damage, its, armour, level, decreases, and, its, interface, shows, it
The separators are used in order to divide the requirement into tokens.
POS TAGGING
POS TAGGED TOKENS
human, unit, gets, damage, armour, level, decreases, interface, shows, ~~when, a, its, and, its, it~~
Those elements that do not provide useful information are removed.
LEMMATIZATION
LEMMA REDUCED TOKENS
human, unit, **get**, damage, armour, level, **decrease**, interface, **show**
The tokens are reduced to their semantic roots.

Figure 1.3: Natural Language Processing Techniques

in terms of substantial information. Lemmatizing analyzes and reduces words, transforming verb tenses such as "decreases" into "decrease".

The same Natural Language Processing techniques used with requirements are applied to code fragments, but include an additional step. This step involves removing stop words, which are programming language reserved words. Once a code fragment is processed, it contains terms such as variable and method names or words that are present in comments. The following example shows a set of two code lines as well as the result once it is processed:

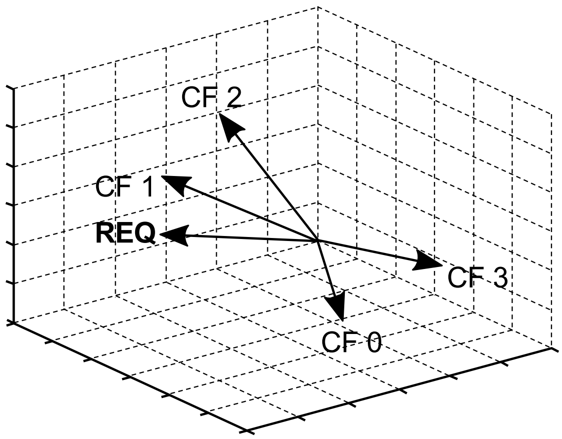
- Code Lines:
  - `3107: int Unit::GetNumberOfWeapons() {`
  - `3108: return moduleWeapons->GetSize();`
- Result:
  - `unit, get, number, weapon,`
    `module, weapon, get, size`

The result obtained illustrates the techniques described above. Tokenizing makes use of separators (e.g., colons or spaces) and other criteria such as naming conventions (CamelCase, in the example) to extract tokens. POS removes elements that are not useful (e.g., "of", a conjunction that does not provide relevant content). Lemma extraction is shown by "weapon", which is a reduced, singular form. Finally, the additional step mentioned above searches for stop words to be filtered. Therefore, terms such as "int" or "return" will be removed since they are both defined as reserved words by the programming language used.

It is assumed that the language used in both the requirement and the code fragment should be similar in order to make the LSI work properly. If those languages are significantly different, not even the Natural Language Processing techniques will prevent the fitness from failing unless the user assists the process manually.

	CODE FRAGMENTS					
TERMS	CF 0	CF 1	CF 2	CF 3	...	REQUIREMENT
HUMAN	0	1	0	0	...	1
UNIT	0	2	0	2	...	1
DAMAGE	0	1	0	0	...	1
DECREASE	0	2	1	0	...	1
ARMOUR	0	2	1	0	...	1
INTERFACE	0	1	1	0	...	1
SHOW	0	0	0	0	...	1
...	...	...	...	...	...	...

LSI RESULTS



RANKING	
CF 1	0.75
CF 2	0.50
CF 0	0.35
CF 3	0.15
...	

Figure 1.4: Fitness Operation via Latent Semantic Indexing (LSI)

1.6.2 Latent Semantic Indexing (LSI) Fitness

After the Natural Language Processing, a co-occurrence matrix is built in order to represent terms in rows and each code fragment in a column, thus providing occurrence counters for every term in each code fragment or in the requirement. In the end, our LSI fitness uses a term set that defines the rows in the co-occurrence matrix, which is a union of two subsets: the terms in the requirement and the terms in the code fragments.

The top part of Fig. 1.4 shows a schematic view of a co-occurrence matrix in our running example. The rows represent the terms from both the code fragments and the requirement. The columns represent the code fragments and the requirement. The values in the cells are the number of occurrences for each term in the code fragments and the requirements.

The co-occurrence matrix must be analyzed in order to elaborate the code-fragment ranking. First, it is normalized and decomposed through Singular Value Decomposition (SVD) [28], which factorizes the matrix and provides a vector set that represents the latent semantic value for every code fragment and the requirement. Similarity is evaluated using the angles formed by such multidimensional vectors, since cosine is a measure of similarity that is 1.0 for identical vectors and 0.0 for orthogonal vectors[36]. In the end, the cosine between each code fragment vector and the requirement vector is a value in the interval [-1, 1] that defines the similarity or proximity to the requirement, which allows a code-fragment ranking to be established.

Let C_1 be a code fragment in the population; let X be the vector representing the latent semantic value of C_1; let Y be the vector representing the latent semantic value of the requirement; the angle between X and Y is θ. The following expression defines the fitness function:

$$fitness(C_1) = \cos(\theta) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} \tag{1.1}$$

The bottom part of Fig. 1.4 shows a three-dimensional graph of the LSI results. The graph shows the representation of each one of the vectors, which are labeled with letters that represent the names of the code fragments. Finally, after the cosines are calculated, a value for each of the code fragments is obtained, indicating its similarity with the requirement.

1.7 Evaluation

Table 1.1: Configurations used in our CODFREL approach and Regular-LSI for the different research questions

	CODFREL	**Regular-LSI**
Research Question 1	Best Code Fragment	Best Complete Method
Research Question 2	Best Code Fragment	10 Best Complete Methods
Research Question 3	Best Code Fragment with Method Granularity	Best Complete Method
Research Question 4	Best Code Fragment with Method Granularity	10 Best Complete Methods
Research Question 5	Best Code Fragment	10 Best Code Lines

This section presents the evaluation of our approach: the research questions, the oracle preparation, the experimental setup, the implementation details and the results obtained.

1.7.1 Research Questions

The following research questions address the evaluation of our approach considering different configurations and how they affect the results obtained. These questions make reference to a threshold, which is the number of complete methods (or code lines, depending on the granularity used) selected as a possible solution by Regular-LSI, the baseline approach. In addition, the research questions take into account the use of code fragments or method granularity by our approach:

RQ$_1$: *Does our CODFREL approach perform better than Regular-LSI when Regular-LSI uses a threshold value (the number of complete methods) of 1?*

RQ$_2$: *Does our CODFREL approach perform better than Regular-LSI if the threshold value is the most widely used?*

RQ$_3$: *Does our CODFREL approach perform better than Regular-LSI if the threshold value is 1 and CODFREL uses method granularity?*

RQ$_4$: *Does our CODFREL approach perform better than Regular-LSI if the threshold value is the most widely used and CODFREL uses method granularity?*

RQ$_5$: *Does our CODFREL approach perform better than Regular-LSI when both use code line granularity (Regular-LSI considers each code line as a document) and the threshold value for Regular-LSI is the most widely used?*

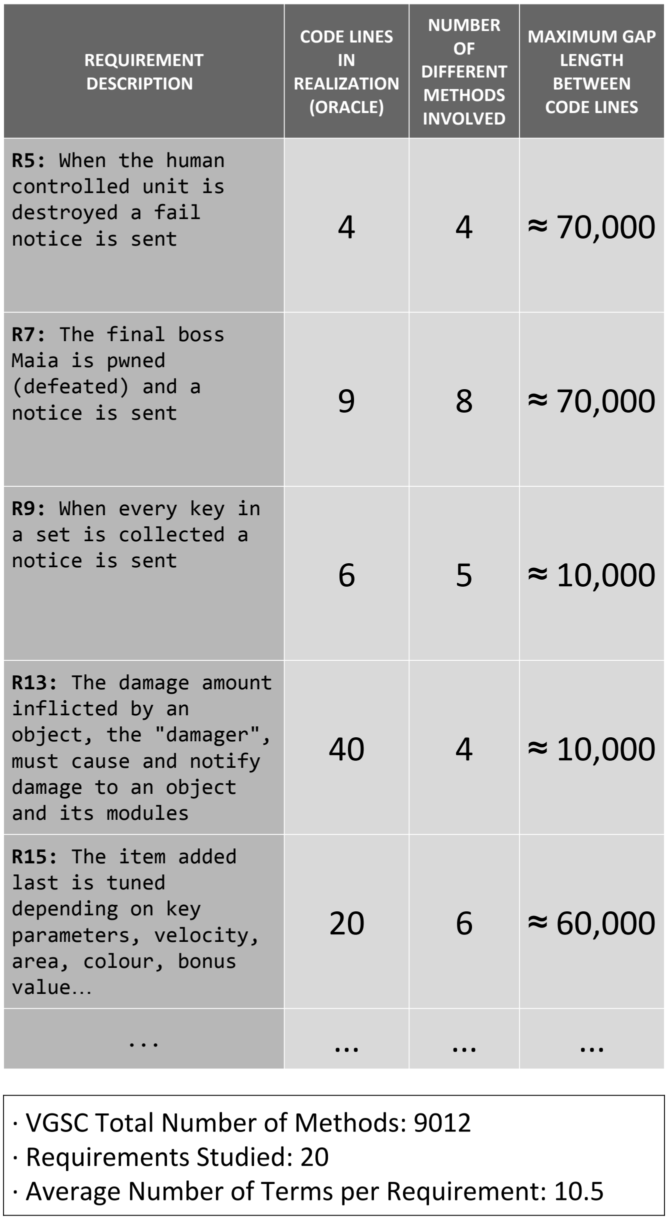
Table 1 shows how the different configurations for our CODFREL approach and Regular-LSI combine according to each research question.

1.7.2 Oracle

The concept of oracle, in the context of our work, is applied to code line sets corresponding to the ground truth or full coverage for a requirement. In other words, this code line set represents the most accurate possible solution corresponding to a requirement. Twenty requirements, as well as the corresponding oracles, are provided by the game development company responsible for the design of Kromaia. Figs. 1.5 and 1.7 show key data regarding the requirement set, although detailed source code information is confidential. The requirements in the set were selected and provided by the developers for being a representative collection in terms of maintenance, and such selection criteria did not depend on dispersion within the VGSC. We perform a fair comparison by configuring the different versions (code line and method granularity) of our approach and Regular-LSI with the same requirement set mentioned, to ensure that neither Regular-LSI nor CODFREL are studied for optimized data sets.

1.7.3 Experimental Setup

The evaluation measures the performance achieved by our approach. In addition, we compare our approach with a baseline approach (Regular-LSI) that achieves the best results in the literature [10]. Regular-LSI selects the method that is most relevant to the requirement by means of LSI. The LSI documents are methods, and the query is the requirement.



REQUIREMENT DESCRIPTION	CODE LINES IN REALIZATION (ORACLE)	NUMBER OF DIFFERENT METHODS INVOLVED	MAXIMUM GAP LENGTH BETWEEN CODE LINES
R5: When the human controlled unit is destroyed a fail notice is sent	4	4	≈ 70,000
R7: The final boss Maia is pwned (defeated) and a notice is sent	9	8	≈ 70,000
R9: When every key in a set is collected a notice is sent	6	5	≈ 10,000
R13: The damage amount inflicted by an object, the "damager", must cause and notify damage to an object and its modules	40	4	≈ 10,000
R15: The item added last is tuned depending on key parameters, velocity, area, colour, bonus value...	20	6	≈ 60,000
...	...	...	...

· VGSC Total Number of Methods: 9012
· Requirements Studied: 20
· Average Number of Terms per Requirement: 10.5

Figure 1.5: Sample containing some of the requirements in the case study and data relative to the methods involved.

The first step involves feeding both our CODFREL approach and Regular-LSI with each of the requirements. Since CODFREL is not a deterministic approach, our approach features 30 runs for each requirement as suggested in [7], whereas Regular-LSI, which is deterministic, features one run.

Once our approach and Regular-LSI finish processing a requirement, they provide the solutions found for this requirement. Our CODFREL approach supplies a code fragment from various code lines that are present in the VGSC. Regular-LSI provides a solution consisting of a set of code lines for a complete method.

The results obtained by our approach and Regular-LSI are compared to the oracle through a confusion matrix, or error matrix [38]. Confusion matrices are used to study the performance achieved by a classification system on a certain test data set. The oracle indicates which data in that set is true or false. Confusion matrices are useful for evaluating the results provided by an approach and the ground truth that the oracle represents.

The confusion matrix arranges the results of the comparison into different categories:

- True Positives (TP), which refer to the number of code lines in the code fragment selected as a solution that are also present in the oracle.
- False Positives (FP), which denote the number of code lines that are present in the proposed solution but are not present in the oracle.
- False Negatives (FN), which denote the number of code lines present in the oracle that are not present in the code fragment or complete method marked as a solution.

Then, performance measurements are derived from the values in the confusion matrix. Specifically, we create a report that includes three performance measurements (precision, recall, and the F-measure).

- **Precision** is the fraction of correct code lines among the code lines selected, according to the corresponding oracle in the result proposed as a solution.

$$Precision = \frac{TP}{TP + FP} \tag{1.2}$$

- **Recall** measures the number of code lines in the oracle that are correctly retrieved over the total number of code lines proposed in that solution.

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

- The **F-measure** corresponds to the harmonic mean of precision and recall. It is used to evaluate accuracy and is defined as follows:

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1.4}$$

1.7.4 Implementation Details

We have used the following libraries to implement the approach of this work: the OpenNLP Toolkit for the Processing of Natural Language Text [6] to develop the Natural Language Processing operations; and the Efficient Java Matrix Library (EJML) [3] to perform LSI and SVD. The computer used in the evaluation was a Toshiba Satellite PRo L830 laptop, with an Intel(R) Core(TM) i5-3317U@1.7GHz processor with 4GB RAM and Windows 8 64-bit.

The fusion operation is applied with a fusion probability (p_f). Through tuning tests, the value of p_f varied changing from preliminary values, like 0.5, to 1 in order to maximize the number of new individuals produced by fusion in each iteration of the algorithm (e.g., the change mentioned would involve not only duplicating the individuals created through fusion, but also duplicating the number of candidates to produce additional, mutated individuals). Additional research could improve parameters like this. The mutation operation is applied with a probability (p_m) of 0.25. The rest of the settings are detailed in Table 2. The focus of this paper is not to tune the values to improve the performance of our approach when applied to a specific problem. As suggested by Arcuri and Fraser [7], default values are good enough to measure the performance of search-based techniques in the context of testing. Nevertheless, we plan to evaluate all of the parameters of our approach in a future work. First, we started with default values used in the literature regarding software model feature traceability [22]. However, since the objective in this work is different (fine-grained requirement traceability in source code) and we use genetic operations to manipulate code fragments, the parameters are the result of starting with those studied in the literature and then performing preliminary tuning experiments. With the current configuration, 7 (μ) parents are combined in pairs to create 21 new code fragments. Apart from those new code fragments and depending on p_m, up to 21 mutated code fragments could also be created. Therefore, it is necessary to discard code fragments from the population after each iteration in order to keep the population stable, with 42 (r) being the maximum number of candidates (those with the lower fitness values) that could be removed.

Table 1.2: CODFREL configuration parameters

Parameter description	**Value**
$Size$: Population size	1000
μ: Number of parents	7
λ: Number of offspring from μ parents	21
r: Maximum number of solutions replaced	
to stabilize population	42
p_f: Fusion probability	1
p_m: Mutation probability	0.25

In general, there are two atomic performance measures for search algorithms: one regarding solution quality and one regarding algorithm speed or search effort. In this paper, we focus on the solution quality.

Therefore, we allocated a fixed amount of wall clock time for each of the runs of our approach. First, we ran some prior tests to determine the time needed to converge, and then we selected the budget time based on those tests. The allocated budget time was 1200 seconds. A prototype of CODFREL can be found at bitbucket.org/svitusj/SCoFBReL

1.7.5 Results

Table 1.3: Precision, Recall, and F-Measure mean values and standard deviations for the case study

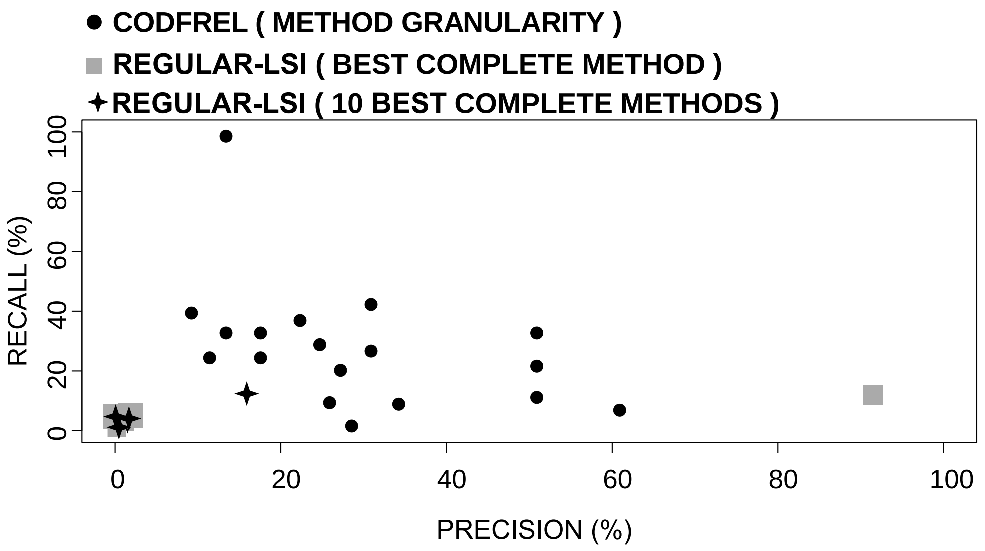
	Precision$\pm$ (σ)	**Recall**$\pm$ (σ)	**F-measure**$\pm$ (σ)
	(%)	(%)	(%)
CODFREL	57±30	27±20	29±13
CODFREL, Method Granularity	28±15	27±20	21±9
Regular-LSI, Best Complete Method	4±19	0.5±2	0.9±4
Regular-LSI, 10 Best Complete Methods	0.7±3	0.5±2	0.6±2
Regular-LSI, 10 Best Code Lines	0.1±0.2	9±11	0.2±0.4

In this subsection, we present the results obtained for the case study in our approach and for Regular-LSI. Fig. 1.6 shows the charts with the recall and precision results. For CODFREL, a dot in the graph represents the average result (after 30 runs and due to random factors) of precision and recall for each of the 20 requirements. In the case of Regular-LSI, the solution for a requirement is deterministic, so there are no repetitions and the dots in the graph represent the results after a single run for each requirement. Table 3 shows the precision, recall, and F-measure mean values obtained for the case study by our approach and Regular-LSI.

1.7.6 Research Question 1

To answer the first research question, it is necessary to study our CODFREL approach with code fragments and a Regular-LSI configuration with a threshold value of 1, since it only selects the best complete method.

RQ$_1$ answer. Fig. 1.6 and Table 1.3 show that CODFREL outperforms Regular-LSI in precision, recall, and F-measure, with average values of



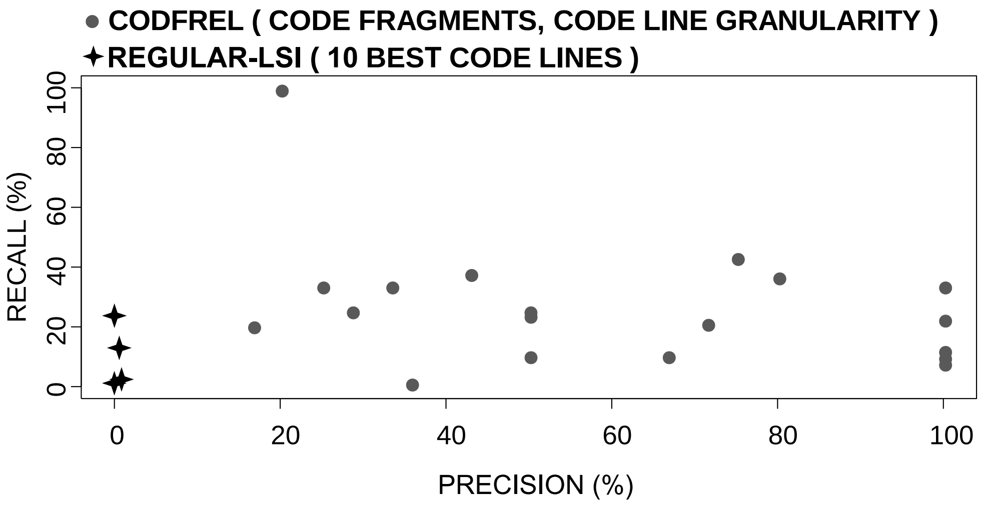


Figure 1.6: Results for our CODFREL approach and Regular-LSI with different configurations

57% in precision, 27% in recall, and 29% in F-measure. Regular-LSI obtained average values of 4%, 0.5%, and 9% in precision, recall, and

F-measure, respectively, for the requirement set studied. The deviations for Regular-LSI, an approach which is deterministic, occur due to performance variations that are related to the different requirements and are not caused by different runs.

1.7.7 Research Question 2

The second research question takes into account that, while human subjects usually do not focus on a single candidate, they tend to consider no more than 10 candidate trace links [10]. In this case, our CODFREL approach uses code fragments and Regular-LSI selects the 10 best complete methods.

RQ$_2$ answer. Fig. 1.6 and Table 1.3 show that CODFREL obtains better results in recall, precision, and F-measure, with values of 27%, 57%, and 29%, respectively. In comparison, Regular-LSI gets significantly lower results for recall (0.5%), precision (0.7%), and F-measure (0.6%).

1.7.8 Research Question 3

The third research question studies a Regular-LSI configuration that only selects the best method and our CODFREL approach with method granularity. This variation implies that CODFREL creates and manipulates code fragments that include every complete method from which code lines were selected, instead of using code line granularity.

RQ$_3$ answer. Fig. 1.6 and Table 1.3 show that CODFREL, with method granularity, outperforms Regular-LSI. Due to the granularity configuration used, precision does not reach 30%, with an average value of 28%, and recall and F-measure values are 27% and 21%, respectively.

1.7.9 Research Question 4

The fourth research question compares the following configurations: our approach CODFREL, with method granularity and Regular-LSI, with a threshold of 10 complete methods.

RQ$_4$ answer. Fig. 1.6 and Table 1.3 show that CODFREL, with method granularity, gets better results than Regular-LSI. Method granularity does have a remarkable impact on precision for our approach, with a value of 28%. However, since the values obtained by Regular-LSI are very low, the difference in the results is high, with this configuration of Regular-LSI selecting the 10 best complete methods as possible solutions.

1.7.10 Research Question 5

The fifth research question considers these configurations: our approach CODFREL with code fragments, and Regular-LSI with code line granularity which involves treating each line as a method using a threshold of 10 code lines.

RQ$_5$ answer. Fig. 1.6 and Table 1.3 show that CODFREL, with code fragments, gets better results than Regular-LSI configured with code line granularity. The granularity used by Regular-LSI significantly affects precision, with a value of 0.1%. Regular-LSI obtains values that are low in recall and in F-measure: 9% and 0.2%, respectively.

1.8 Discussion

The requirements were provided by one of the developers of Kromaia before we started this research work. The requirement selection criteria used by the developer did not depend on the requirement implementation being condensed in just one complete method or featuring a high dispersion level. In fact, the criteria involved the selection of requirements that were relevant in terms of maintenance. Most of the requirements provided were dispersed, with the average number of methods for a dispersed requirement being around 5.3. Fig. 1.5 shows that, even for requirements that include less than five lines, according to the oracles, it is not uncommon to find four or more methods involved. In addition, Fig. 1.7 shows that, in terms of cohesion, the realizations of the requirements studied include gaps between code lines of up to 70,000 code lines. This suggests that dispersed requirements should not be neglected in maintenance tasks.

Regular-LSI, which was selected for currently being the best performing IR technique, does not work properly for highly dispersed requirements in the VGSC of the case study, and the average results obtained are not good in the performance measures studied. The main cause for this is the fact that Regular-LSI works in terms of complete methods to trace requirements that are dispersed in various methods. Besides that, in those cases featuring a method that combines code lines that are relevant to the requisite with code lines that are not relevant, the irrelevant code lines prevent the method from getting a higher score. For instance, in the following requirement:

- `R1: When a human unit gets damage, its armour level decreases and its interface shows it.`

There is an event (gets damage) that has a noticeable impact on an entity known as a human unit in different contexts: the internal logics and structure in the entity, which need to be modified; and the interface elements that are directly related to this entity, which provide visual feedback regarding its internal status. This requisite is dispersed in five methods, each one of which only features an average of 13% (maximum 33%; minimum 8%) of relevant code lines for R1.

Our CODFREL approach outperforms Regular-LSI thanks to the fact that it uses code fragments. The total number of possible code fragments ($2^{145,000}$) makes a thorough exploration and evaluation unfeasible. The use of an evolutionary algorithm, however, allows our approach to explore the search space, and it gets better results than Regular-LSI that is configured to work with code line granularity, due to the size of the search space and high requirement dispersion. The results show that it is possible to use the proposed encoding and genetic operations in commercial software products that are similar to Kromaia. In each iteration of the evolutionary algorithm, the new code fragments created with the fusion operator are progressively larger, but code fragments that are remarkably large are prone to being discarded if the code lines accumulated do not contribute to increasing the value given to those code fragments.

There is another factor to be taken into account regarding the results: the use of keywords. There are terms featured in requirements, which, in spite of being relevant for such requirements, are ambiguous. In the requirement R1, terms like "decrease", "armour", and "level" are relevant for the requirement. "Level", however, is widely used and accepted as a video game Domain-Specific Language term with different but equally valid meanings: "level" could be considered as a stage or zone that should be cleared by the player, but it could also refer to the current player status. For that reason, "level" is relevant but ambiguous.

However, terms that are relevant but ambiguous are not discarded in our approach since they are used (along with the keywords) to calculate fitness values. In contrast, since keywords are terms that are both relevant and unambiguous, they are given more importance in our approach. They are not only used in fitness calculations but are also used to provide guidance in additive mutation. In R1, "decrease" and "armour" are suitable keywords that comprise the main concepts involved by the requirement. Additionally, we have considered the effects of guidance in subtractive mutation, which is an operator that, in our approach, removes code lines by randomly selecting them. We included a modified operator that takes keywords into account, like additive mutation, and tends to preserve code lines that include keywords. One possible disadvantage of this alternative operator is the low probability of removing code lines that contain keywords, even if such lines are not relevant; therefore, solution candidates with such content could not be improved through random modifications. In comparison to the results obtained with the first version of the operator, precision and recall increased an average of 20% in four of the 20 requirements studied. However, for 15% of the requirements the average results were 16% lower after using the new operator. This data could be studied in future works to produce a better subtractive mutation operator.

Even if CODFREL outperforms Regular-LSI, it does not achieve solutions that include every code line that is relevant for the requirements.

Our analysis of the results suggests that tacit knowledge has a negative effect on the results.

Tacit knowledge is often assumed to be known by every domain expert. This assumption leads to a lack of documented presence of that knowledge, and requirements are no exception. The tacit knowledge related to the domain involved by requirements is usually considered and shared by the developers, who are responsible for the VGSC as well as for providing the requirements. In the end, every aspect of the domain knowledge (including tacit knowledge) that remains unwritten and the information provided by requirements are present in the VGSC. Therefore, requirements that do not feature a detailed description that reflects all relevant knowledge are incomplete, and the approaches searching for solutions will experience difficulties trying to find optimal results. The following example is a requirement that omits tacit knowledge that is actually present in the VGSC and the solution to be found:

- `R2: Shot input makes the human unit fire projectiles.`

This requirement omits relevant information regarding the modular nature of the VGSC featured by many entities. Units contain weaponry modules, which contain weapons. Besides, weapons internally manage a variable number of cannons, but only the cannons marked as "valid" are those responsible for ultimately firing projectiles.

Since tacit knowledge is the main issue to be studied in order to achieve better results with our approach, we plan to research it in more depth in our future works. In order to address this subject, we intend to use reformulation techniques so as to expand the requirements with elaborated descriptions provided by domain experts.

1.9 Future Work

The main issues to be addressed in future works are the eventual upgrade of genetic operators, the re-evaluation of parameters, and research on tacit knowledge:

With regard to genetic operators, it is possible that mutations involving the removal of code lines from code fragments should be guided, like additive mutations. In order to include such an operation, the management of terms that are not relevant, as opposed to keywords, could be useful.

The use of different values for the collection of parameters used in our approach could be studied in future works since tuning those values, like the fusion probability, the mutation probability, or the number of code fragments selected to be the parents of the next generation, could lead to configurations having an impact on the results and the time used. Due to the time required to test different configurations, future works should focus on this issue in particular.

Tacit knowledge is a key issue that should also be considered in future works. It would be necessary to expand the requirements available, and domain experts would be required for that task. The direct participation of domain experts would play a key role in providing additional explicit knowledge to be used in the guidance of the evolutionary algorithm in our CODFREL approach.

1.10 Threats to Validity

The classification for possible threats to validity in [43] reflects the necessary awareness regarding the limitations of our approach. This classification covers aspects that are related to both the approach itself and the commercial video game case study:

- **Internal Validity** refers to eventual issues related to existing causal relations. There is a possible risk for code line selection to be biased during the code fragment creation process due to the use of keywords in the population initialization as well as in guided additive mutations. That risk is reduced since our approach includes random deviation measures in these selection processes.
- **External Validity** is mainly concerned with the actual extent to which the results found can be generalized to case studies that are different from Kromaia, which is the commercial video game case

study presented in this work. Two factors that increase the possibilities of generalization in the VGSC are the extensive use of a strict coding style and design patterns. These are widely used in real-time applications that are similar to video games. Nevertheless, our results should be replicated with other case studies before assuring their generalization.

- **Construct Validity** is an aspect worth taking into consideration since there is a risk that the operations involved in our approach may not accurately represent the desired functionalities for this research. The use of widely accepted measures such as precision and recall minimizes the risk described.
- **Reliability** deals with the possibility of the researchers influencing both the analysis process and the data used in the approach. In order to minimize the knowledge regarding the commercial video game case study, the VGSC comprehends a vast number of code lines, thus preventing requirements from being located too easily.

1.11 Related Work

A recent traceability survey [10] has identified the need for more industrial case studies. This work also shows that in spite of being the most commonly used, algebraic models (LSI and Vector Space Model) search in solution spaces with sizes below 2^{500}. Our work deals with an industrial case study that features a significantly wide space ($2^{145,000}$) to be explored and dispersed requirements, as shown in Fig. 1.7. Therefore, we work with code fragments instead of complete methods, and we search for possible solutions by means of an evolutionary algorithm since this type of algorithm has proven to be useful in large search spaces [**Arcega2018**].

There are works that use design documents or domain models to support traceability [13] [27]. However, in the case study in this work, those artifacts were not available. Currently, video game developers are pressured by what is called "the age of crunch" [1] and the ever-increasing high demand of game content, which is caused by early access releases, post-launch updated versions, DLC (Downloadable Content), and games as a service. In this context, these artifacts end up not being synchronized or they are not even created, so approaches like CODFREL, which work when the artifacts mentioned are not available, are necessary.

The ADAMS document management system by De Lucia et al. [15] was used as evaluation context in trace recovery empirical experiments through LSI. Through case studies with students, as well as different controlled experiments, they have reinforced the empirical basis (De Lucia et al. [16], [18], [17]). Also, various studies by Cleland-Huang and colleagues, which are focused on Information Retrieval (IR)-based trace recovery, show the use of PIN-based retrieval as a model that supports the introduction of probabilistic trace recovery. This model was implemented in their tool, Poirot (Lin et al. [29]). To a great extent, their work focuses on accuracy improvements for their tool. The enhancements include the localization of key phrases [45], synonymy management with a thesaurus, and a glossary that weights the most important terms in the project with higher values [45].

In comparison to the works mentioned above, our work makes use of LSI to guide an evolutionary algorithm. LSI does not give values directly to complete artifacts such as methods but rather guides the exploration within the solution space.

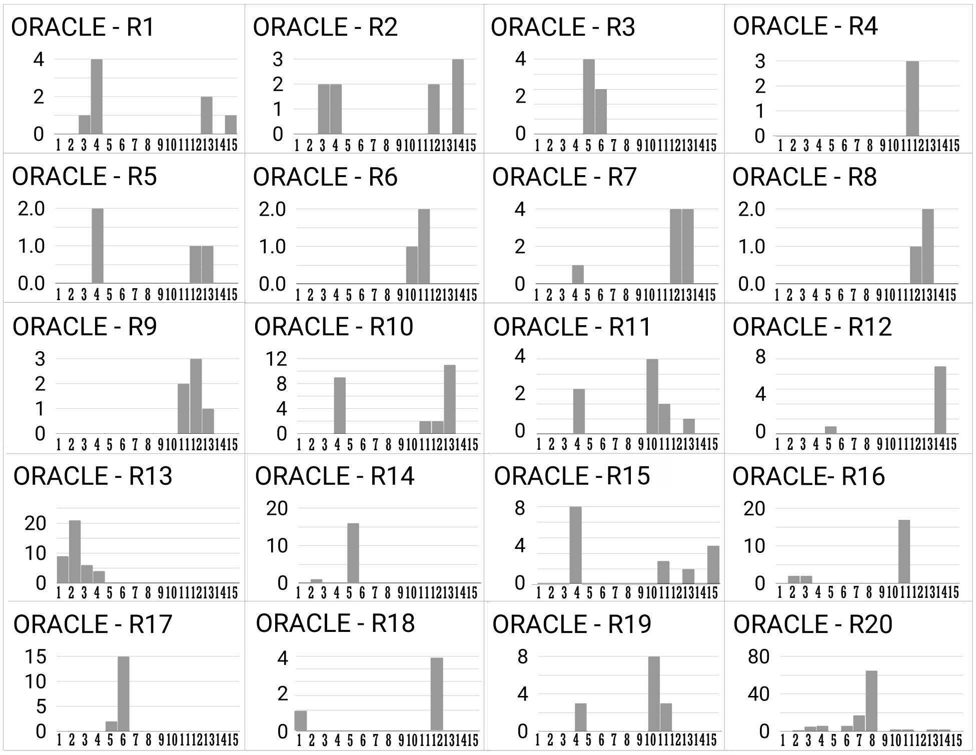
The use of probabilistic models has been used by several researchers to support trace recovery. For instance, the probabilistic topic model Latent Dirichlet Allocation is one of the various IR models combined by Dekhtyar et al. [19] using a voting scheme. Abadi et al. [2] proposed using Probabilistic Latent Semantic Indexing and two information theory-based concepts: Jensen-Shannon Divergence and Sufficient Dimensionality Reduction. Parvathy et al. [31] suggested the Correlated Topic Model, while Getters et al. [23] proposed the Relational Topic Model. The use of a non-centralized set of self-organized agents that work together to extract conclusions is a swarm-like approach to trace recovery implementation proposed by Sultanov and Huffman Hayes [39]. Similarity calculations in the Cartesian plane are also possible. Capobianco et al. [12] suggested using B-Splines as Natural Language artifact representations so that similarity would be given by the distance between these splines on the Cartesian plane.

Unlike those works, our work searches for solutions in terms of code fragments. These solutions could be any code line set in the source code. Solutions of this kind are necessary in commercial contexts like the AAA video game industry, where requirements are dispersed across several methods.

Genetic Algorithms have been used to configure and assemble IR processes automatically in order to support different software engineering tasks [20], thereby positively affecting the time and resources spent on maintenance. These approaches determine near optimal solutions for the different IR process stages without training, and they outperform previous approaches without remarkable differences in comparison to combinatorial and supervised approaches. Our work also uses an evolutionary algorithm, but it explores in a completely different way. Instead of using the evolutionary algorithm to search for the best IR technique combination, we use the evolutionary algorithm to search the vast code fragment solution space, which is the case for AAA commercial video game software ($2^{145,000}$, in our case study).

Other works have studied the extent to which IR techniques can provide decision support in the context of large, industrial software engineering tasks in terms of traceability and maintenance [40]. In these works, the researchers noticed issues regarding the difficulties of scaling IR techniques to industry data, due to latent semantic analysis. The way in which IR-based traceability recovery tools are used by developers and how they validate/discard correct information and false positives has been studied in works that are focused on going beyond the performance analysis of IR-based traceability methods [9]. The approach used in those works suggests counting recovered traceability links in order to increase the quality of the validation process that is carried out by the users that are working with recovery tools. Our approach not only outperforms Regular-LSI, but it also finds a new cause behind the inability to obtain better requirement traceability results: tacit knowledge that is not formalized when the requirements are written.

Recent works [26, 44, 5] propose the use of Neural Networks to address the challenge of traceability. Guo et al. [26] leverage Word Embedding and Recurrent Neural Network (RNN) models to generate trace links, which



· VGSC Total number of code lines: 145,914

· Horizontal Axis: VGSC, divided in 15 equal regions (10,000 contiguous code lines per region)

· Vertical Axis: Number of code lines in the requirement realization that are located in each region

Figure 1.7: Dispersion of the requirement realizations in the source code of the case study.

contain the requirements artifact semantics and the domain knowledge. Zhao et al. [44] propose training deep neural networks to generate text-based knowledge in software repositories in order to improve the accuracy of TLR. The work in [5] presents some challenges in traceability and some of their proposals consider addressing these traceability issues through neural networks.

The above works based on Neural Networks require the existence of a knowledge base for training. For example, in [26], the training set is composed of 45% of the 769,366 artifacts, so this training set contains 423,151 feature vectors. However, some industrial companies do not store enough information to create the required knowledge base for Neural Networks. Actually, this lack of documented knowledge has been previously reported as the knowledge vaporization problem [41]. Nevertheless, these domains also need to recover the traceability links, and our CODFREL approach as well as the Regular-LSI approach can be applied for traceability recovery even without a knowledge base.

1.12 Conclusions

In comparison to those generated by Regular-LSI, the solutions provided by our approach are better starting points, assuming, however, that both approaches need to be refined manually. The use of evolutionary algorithms and code fragments improves the results obtained by Regular-LSI, since a VGSC like the one featured in the video game case study involves highly dispersed methods and a huge solution space. However, tacit knowledge, which is not explicitly present in the requirements, prevents our approach from achieving better solutions. Getting domain experts to be more involved could make this implicit knowledge become explicit. To facilitate the adoption of CODFREL, we have made a reference implementation freely available.

Acknowledgements

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).

Bibliography

[1] IGDA, International Game Developers Association, 2018 (cit. on p. 58).

[2] A. Abadi, M. Nisenson, and Y. Simionovici. “A Traceability Technique for Specifications”. In: *Proc. 16th IEEE Int. Conf. Program Comprehension.* June 2008, pp. 103–112. DOI: `10.1109/ICPC.2008.30` (cit. on p. 59).

[3] Peter Abeles. *Efficient Java Matrix Library.* `http://ejml.org/`. [Online; accessed 9-November-2017]. 2017 (cit. on p. 48).

[4] Michael Affenzeller et al. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications.* 1st. Chapman & Hall/CRC, 2009. ISBN: 1584886293, 9781584886297 (cit. on p. 35).

[5] Giuliano Antoniol et al. “Challenges of Traceability: The Next Ten Years”. In: 2017 (cit. on pp. 60, 61).

[6] *Apache OpenNLP: Toolkit for the processing of natural language text.* `https://opennlp.apache.org/`. [Online; accessed 12-November-2017]. 2017 (cit. on p. 48).

[7] Andrea Arcuri and Gordon Fraser. “Parameter tuning or default values? An empirical investigation in search-based software engineering”. In: *Empirical Software Engineering* 18.3 (2013), pp. 594–623. ISSN: 1573-7616. DOI: `10.1007/s10664-013-9249-9` (cit. on pp. 47, 49).

[8] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search.* 2nd. USA: Addison-Wesley Publishing Company, 2008, pp. 134–144. ISBN: 9780321416919 (cit. on p. 29).

[9] Gabriele Bavota et al. “Enhancing software artefact traceability recovery processes with link count information”. In: *Information and Software Technology* 56.2 (2014), pp. 163–182. ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2013.08.004` (cit. on p. 60).

[10] Markus Borg, Per Runeson, and Anders Ardö. "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability". In: *Empirical Software Engineering* 19.6 (Dec. 1, 2014), p. 1565. ISSN: 1573-7616. DOI: `10.1007/s10664-013-9255-y` (cit. on pp. 28, 30, 45, 52, 58).

[11] G. Capobianco et al. "On the role of the nouns in IR-based traceability recovery". In: *Proc. IEEE 17th Int. Conf. Program Comprehension.* May 2009, pp. 148–157. DOI: `10.1109/ICPC.2009.5090038` (cit. on p. 39).

[12] G. Capobianco et al. "Traceability Recovery Using Numerical Analysis". In: *Proc. 16th Working Conf. Reverse Engineering.* Oct. 2009, pp. 195–204. DOI: `10.1109/WCRE.2009.14` (cit. on p. 59).

[13] Jane Cleland-Huang, Jane Huffman Hayes, and J.M. Domel. "Model-based traceability". In: May 2009, pp. 6–10. ISBN: 978-1-4244-3741-2. DOI: `10.1109/TEFSE.2009.5069575` (cit. on p. 58).

[14] Lawrence Davis. *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991 (cit. on p. 35).

[15] A. De Lucia et al. "ADAMS Re-Trace: A Traceability Recovery Tool". In: *Proc. Ninth European Conf. Software Maintenance and Reengineering.* Mar. 2005, pp. 32–41. DOI: `10.1109/CSMR.2005.7` (cit. on p. 59).

[16] A. De Lucia et al. "COCONUT: COde COmprehension Nurturant Using Traceability". In: *Proc. 22nd IEEE Int. Conf. Software Maintenance.* Sept. 2006, pp. 274–275. DOI: `10.1109/ICSM.2006.19` (cit. on p. 59).

[17] Andrea De Lucia, Rocco Oliveto, and Genoveffa Tortora. "Assessing IR-based traceability recovery tools through controlled experiments". In: *Empirical Software Engineering* 14.1 (2009), pp. 57–92. ISSN: 1573-7616. DOI: `10.1007/s10664-008-9090-8` (cit. on p. 59).

[18] Andrea De Lucia et al. "Recovering Traceability Links in Software Artifact Management Systems Using Information Retrieval Methods". In: *ACM Trans. Softw. Eng. Methodol.* 16.4 (Sept. 2007). ISSN: 1049-331X. DOI: `10.1145/1276933.1276934` (cit. on p. 59).

[19] A. Dekhtyar et al. "Technique Integration for Requirements Assessment". In: *Proc. 15th IEEE Int. Requirements Engineering Conf. (RE 2007).* Oct. 2007, pp. 141–150. DOI: `10.1109/RE.2007.17` (cit. on p. 59).

[20] B. Dit. "Configuring and Assembling Information Retrieval Based Solutions for Software Engineering Tasks". In: *Proc. IEEE Int. Conf. Software Maintenance and Evolution (ICSME).* Oct. 2016, pp. 641–646. DOI: `10.1109/ICSME.2016.85` (cit. on p. 60).

[21] Bogdan Dit et al. "Feature location in source code: a taxonomy and survey". In: *Journal of Software: Evolution and Process* 25.1 (2013), pp. 53–95. ISSN: 2047-7481. DOI: `10.1002/smr.567` (cit. on p. 38).

[22] Jaime Font et al. "Leveraging variability modeling to address metamodel revisions in Model-based Software Product Lines". English. In: *Computer Languages, Systems & Structures* 48.Complete (2017), pp. 20–38. DOI: `10.1016/j.cl.2016.08.003` (cit. on p. 49).

[23] M. Gethers et al. "On integrating orthogonal information retrieval methods to improve traceability recovery". In: *Proc. 27th IEEE Int. Conf. Software Maintenance (ICSM).* Sept. 2011, pp. 133–142. DOI: `10.1109/ICSM.2011.6080780` (cit. on p. 59).

[24] Arbi Ghazarian. "A Research Agenda for Software Reliability". In: *IEEE Reliability Society 2009 Annual Technology Report* (2010) (cit. on p. 28).

[25] O. C. Z. Gotel and C. W. Finkelstein. "An analysis of the Requirements Traceability Problem". In: *Proc. IEEE Int. Conf. Requirements Engineering.* Apr. 1994, pp. 94–101. DOI: `10.1109/ICRE.1994.292398` (cit. on p. 28).

[26] Jin Guo, Jinghui Cheng, and Jane Cleland-Huang. "Semantically Enhanced Software Traceability Using Deep Learning Techniques". In: *Proceedings of the 39th International Conference on Software Engineering.* ICSE '17. Buenos Aires, Argentina: IEEE Press, 2017, pp. 3–14. ISBN: 978-1-5386-3868-2. DOI: `10.1109/ICSE.2017.9` (cit. on pp. 60, 62).

[27] Jin Guo et al. "Towards an Intelligent Domain-specific Traceability Solution". In: *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering.* ASE '14. Vasteras, Sweden: ACM, 2014,

pp. 755–766. ISBN: 978-1-4503-3013-8. DOI: 10.1145/2642937.2642970 (cit. on p. 58).

[28] Thomas K Landauer, Peter W Foltz, and Darrell Laham. "An introduction to latent semantic analysis". In: *Discourse processes* 25.2-3 (1998), pp. 259–284 (cit. on pp. 28, 38, 43).

[29] J. Lin et al. "Poirot: A Distributed Tool Supporting Enterprise-Wide Automated Traceability". In: *Proc. 14th IEEE Int. Requirements Engineering Conf. (RE'06).* Sept. 2006, pp. 363–364. DOI: 10.1109/RE.2006.48 (cit. on p. 59).

[30] Dapeng Liu et al. "Feature Location via Information Retrieval Based Filtering of a Single Scenario Execution Trace". In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering.* ASE '07. Atlanta, Georgia, USA: ACM, 2007, pp. 234–243. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321667 (cit. on p. 38).

[31] Anju G. Parvathy, Bintu G. Vasudevan, and Rajesh Balakrishnan. "A Comparative Study of Document Correlation Techniques for Traceability Analysis". In: *ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume ISAS-2, Barcelona, Spain, June 12-16, 2008.* 2008, pp. 64–69 (cit. on p. 59).

[32] Denys Poshyvanyk et al. "Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval". In: *IEEE Transactions on Software Engineering* 33.6 (June 2007), pp. 420–432. ISSN: 0098-5589. DOI: 10.1109/TSE.2007.1016 (cit. on p. 38).

[33] P. Rempel and P. Mäder. "Preventing Defects: The Impact of Requirements Traceability Completeness on Software Quality". In: *IEEE Transactions on Software Engineering* 43.8 (Aug. 2017), pp. 777–797. ISSN: 0098-5589. DOI: 10.1109/TSE.2016.2622264 (cit. on p. 28).

[34] Meghan Revelle, Bogdan Dit, and Denys Poshyvanyk. "Using Data Fusion and Web Mining to Support Feature Location in Software." In: *ICPC.* 2010, pp. 14–23. ISBN: 978-0-7695-4113-6 (cit. on p. 38).

[35] Julia Rubin and Marsha Chechik. "A survey of feature location techniques". In: *Domain Engineering*. Springer, 2013, pp. 29–58 (cit. on p. 39).

[36] Amit Singhal et al. "Modern information retrieval: A brief overview". In: *IEEE Data Eng. Bull.* 24.4 (2001), pp. 35–43 (cit. on p. 43).

[37] George Spanoudakis and Andrea Zisman. "Software Traceability: a Roadmap". In: *Handbook Of Software Engineering And Knowledge Engineering: Vol 3: Recent Advances*. World Scientific, 2005, pp. 395–428 (cit. on p. 28).

[38] Stephen V. Stehman. "Selecting and interpreting measures of thematic classification accuracy". In: *Remote Sensing of Environment* 62.1 (1997), pp. 77–89. ISSN: 0034-4257. DOI: `http://dx.doi.org/10.1016/S0034-4257(97)00083-7` (cit. on p. 47).

[39] H. Sultanov and J. H. Hayes. "Application of Swarm Techniques to Requirements Engineering: Requirements Tracing". In: *Proc. 18th IEEE Int. Requirements Engineering Conf.* Sept. 2010, pp. 211–220. DOI: `10.1109/RE.2010.33` (cit. on p. 59).

[40] Michael Unterkalmsteiner et al. "Large-scale information retrieval in software engineering - an experience report from industrial application". In: *Empirical Software Engineering* 21.6 (Dec. 1, 2016), p. 2324. ISSN: 1573-7616. DOI: `10.1007/s10664-015-9410-8` (cit. on p. 60).

[41] Jan Salvador van der Ven et al. "Design Decisions : The Bridge between Rationale and Architecture". In: 2006 (cit. on p. 62).

[42] R. Watkins and M. Neal. "Why and How of Requirements Tracing". In: *IEEE Software* 11.4 (July 1994), pp. 104–106. ISSN: 0740-7459. DOI: `10.1109/52.300100` (cit. on p. 28).

[43] Claes Wohlin et al. *Experimentation in software engineering*. 2012 (cit. on p. 57).

[44] Yu Zhao et al. "Using Deep Learning to Improve the Accuracy of Requirements to Code Traceability". In: *Challenges of Traceability: The Next Ten Years*. 2017, pp. 22–24 (cit. on pp. 60, 61).

[45] Xuchang Zou, Raffaella Settimi, and Jane Cleland-Huang. "Improving automated requirements trace retrieval: a study of term-based enhancement methods". In: *Empirical Software Engineering* 15.2 (2010), pp. 119–146. ISSN: 1573-7616. DOI: `10.1007/s10664-009-9114-z` (cit. on p. 59).

Chapter 2

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

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 b osses). I n t his w ork, w e p resent 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 h ours o f u nattended t ime i n c omparison to ten months of work by the developers. We perform a statistical analysis, and we make an implementation of EMoGen readily available.

2.1 Introduction

Game Software Engineering (GSE) is a research area that was compared with classic Software Engineering for the first time by McShaffry in 2003 [43]. A recent survey, published in 2010, showed an overview of GSE research works and described the increasing interest in GSE [3]. 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 [52]. In the following years, subsequent works have continued focusing on the generation of source code from models [46, 47, 60, 73].

Some of the above MDE works [52, 73] use UML as the modelling language, one MDE work [60] uses process models, while the rest of them [46, 47] use Domain-Specific Modelling Languages (DSL). The major advantage 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 [23] or Unity [65], and are much closer to the problem domain (the content of the video game) related to most popular programming languages (e.g., C++) [56]. 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.

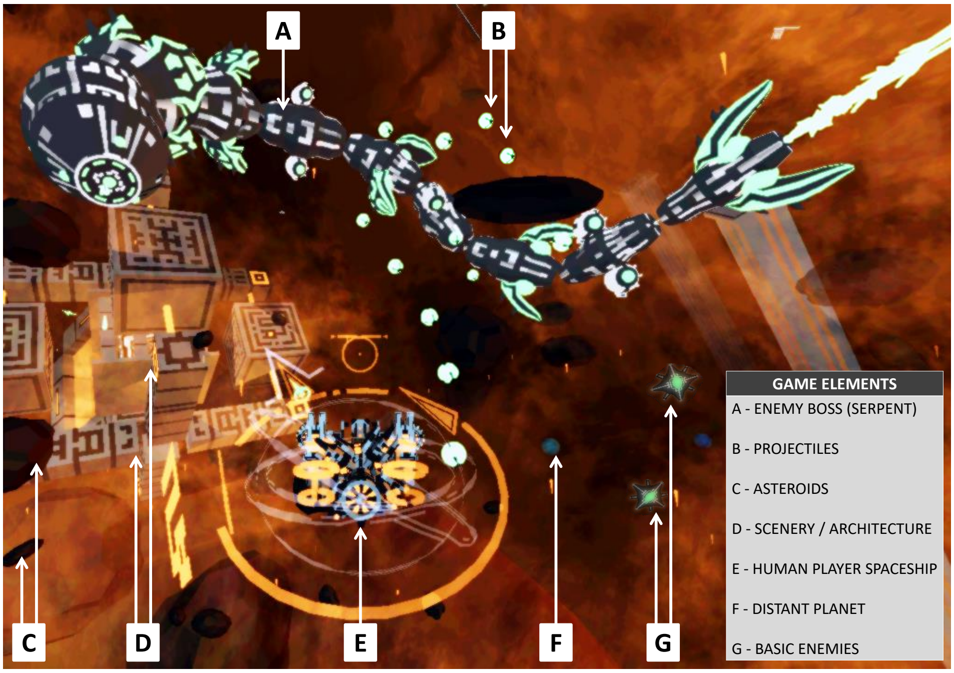


Figure 2.1: Screenshot showing game content in Kromaia.

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 2.1 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 2.1, G): ships and creatures that try to damage the player unit by firing projectiles that damage the player spaceship (Figure 2.1, 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 2.1, 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:

- 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 3.1 depicts an excerpt of the SDML that specifies one of the bosses of Kromaia (see the bottom of Figure 3.1). 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 3.1 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 [14] has proven to outperform more sophisticated algorithms, and is a common sanity check practice in the Search-based Software Engineering [30] 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 [38], 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 [1]. 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 2.2 summarizes related works. Section 2.3 presents Model-Driven Engineering for Video Games. Section 2.4 describes our EMoGen approach. Section 3.5 deals with evaluation. Section 2.6 presents the discussion. Section 2.7 describes the threats to validity, and Section 8 presents the conclusion of the paper.

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

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

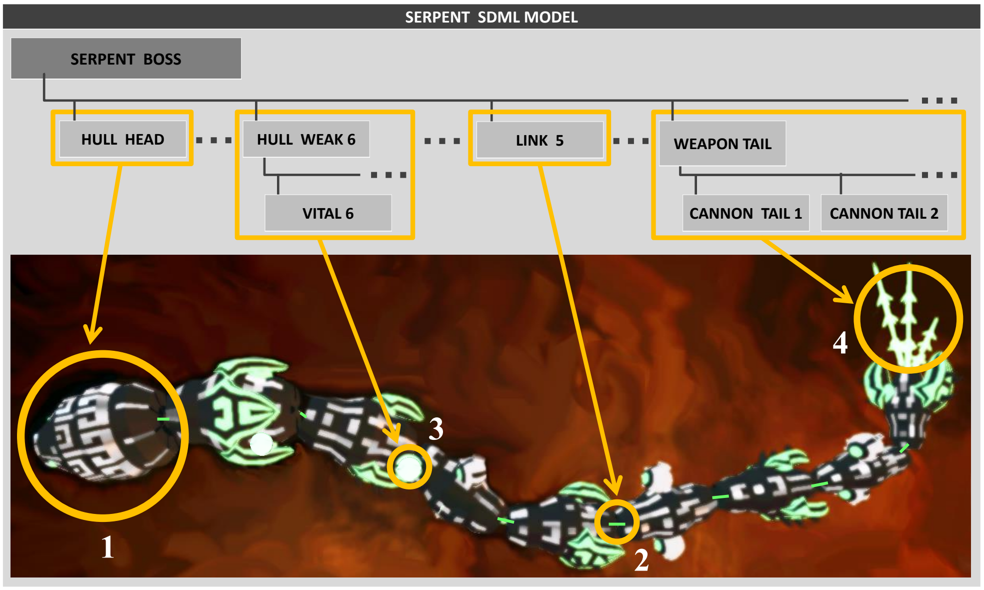


Figure 2.2: SDML model of a boss (top) and the boss at run-time (bottom).

2.2 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.

2.2.1 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 [3] identified only one work that applied Model-Driven Development to video games [52]. 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. [46, 47] presents model-driven approaches that are intended to minimize errors, time, and cost in multi-platform video game development. The work in [46] proposes a Domains-Specific Language, named Gade4all, and focuses on tablet and smartphone-oriented games. Solís-Martínez et al. [60] 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. [73] 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 [65] and Unreal Engine [23] 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. [76] 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 interac-

[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.

tion of the characters, variation in the properties, etc.). However, the case study is a simplified text game. In addition, [76] 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 [28], 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 [28] 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.

2.2.2 Procedural Content Generation

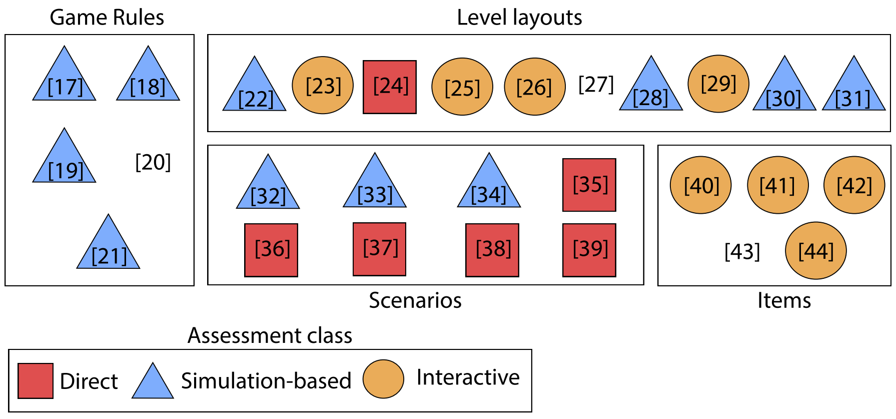


Figure 2.3: Overview of the PCG related work.

Figure 2.3 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 [33, 69, 18, 59, 55]. 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. [67, 49, 61, 35, 62, 74, 54, 51, 45, 40]

Scenarios. These cover the structure of both puzzles [48, 9, 58] and maps/terrains [11, 22, 68, 41, 10]. 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 [31, 42, 32, 72, 53] 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 2.3, we classify the works in relation to their type of assessment following the Togelius' classification. Togelius et al. [70] classified assessment as direct, simulation-based, or interactive. Direct assessment is depicted as a red square in Figure 2.3 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 [74, 64] or they are not explained [59, 72]. They are represented in Figure 2.3 without a geometric classification.

2.2.3 Analysis of the research gap

There is a trend at the intersection of MDE and game software engineering (see subsection 2.2.1) 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 2.2.2), 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 mechanics, 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.

2.3 Model-Driven Engineering for Video Games: The Kromaia Case

This section gives background on the role of models in Kromaia. Model-Driven Engineering (MDE) [37] 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) [44], 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 2.4. 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 2.4).

The bottom of Figure 3.1 shows a final boss that is included in the video game case study. Examples of the SDML concepts used in bosses are the following:

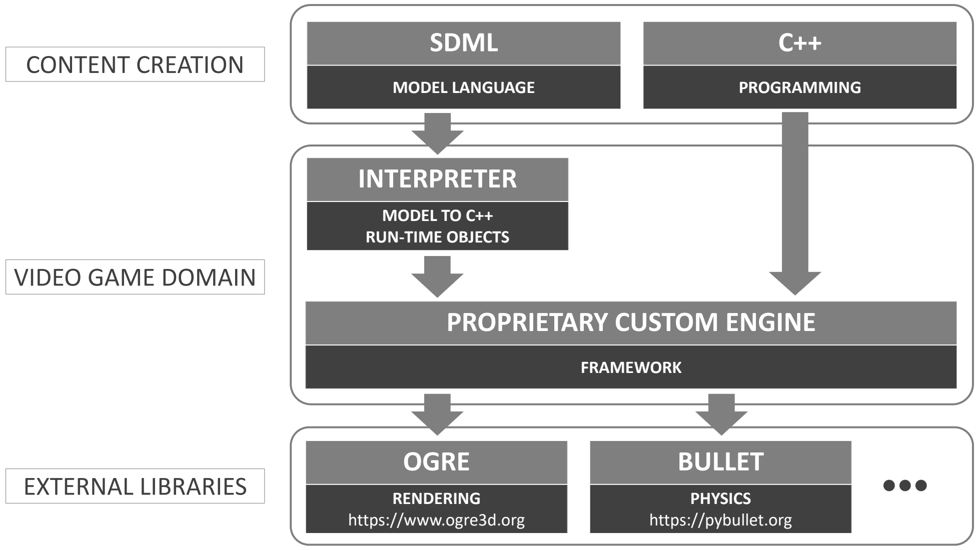


Figure 2.4: The different architecture layers in Kromaia, the video game case study.

Hulls and Links: Hulls (see circle 1 of Figure 3.1) 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 3.1). 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 3.1) 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 3.1) 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 3.1. 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 EMoGen approach relies on an evolutionary algorithm to explore the search space.

2.4 Our EMoGen Approach

This section presents our EMoGen approach, which leverages evolutionary computation to generate human-competitive software models. Figure 3.2 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 2.3).

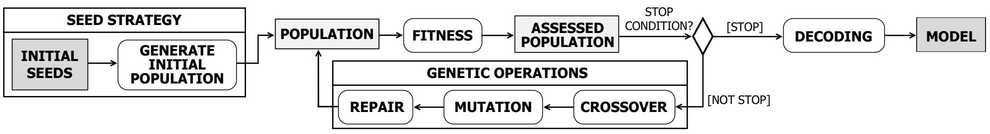


Figure 2.5: Overview of the EMoGen Approach.

2.4.1 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 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 2.3). 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:

$$clamp_{[0,1]}(x) = max(0, min(x, 1)) \tag{2.1}$$

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) ($V_{Optimal}$):

$$F_{Victory} = clamp_{[0,1]}\left(1 - \frac{\mid V_{Optimal} - V_P \mid}{V_{Optimal}}\right) \tag{2.2}$$

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):

$$F_{Health} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{V_P} \frac{|\Theta_{Optimal} - \Theta_P|}{\Theta_{Optimal}}}{V_P}\right) \tag{2.3}$$

$F_{Overall}$ 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):

$$F_{Overall} = min(Validity, \frac{\sum_{i=1}^{N} F_i}{N}) \tag{2.4}$$

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.

2.4.2 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 [21, 4, 5] 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 3.4 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 3.1 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 3.3 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

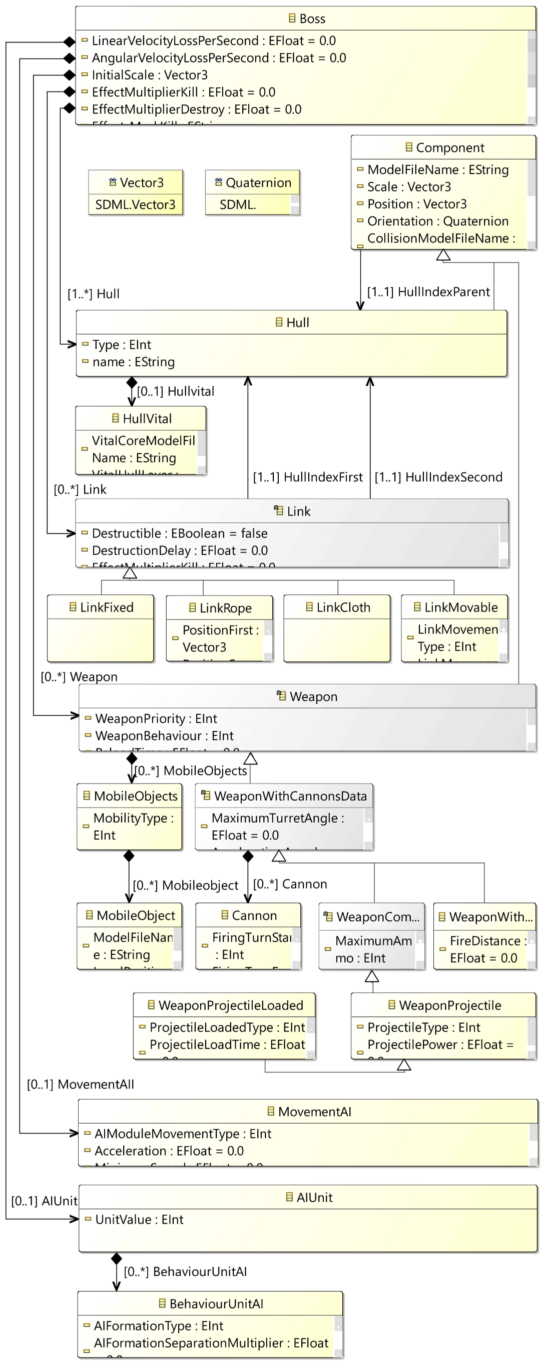
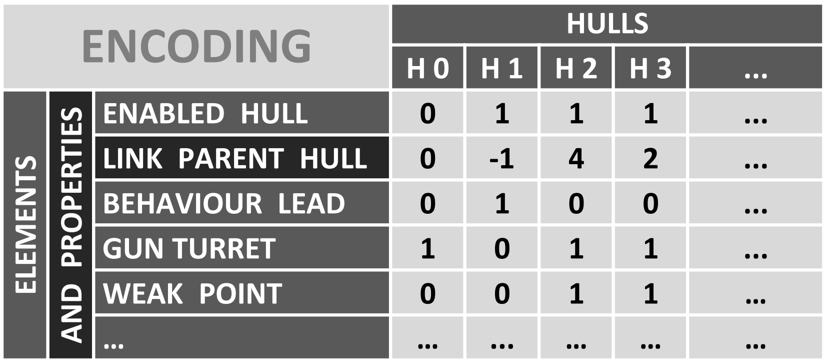


Figure 2.6: Excerpt of the Metamodel of the boss models.



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 2.7: Example of encoding for boss models.

in Figure 3.3, `Hull0` would not be present in the model and `Hull2` would have a turret.

Properties: The second row in Figure 3.3 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 3.3 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 3.3.

2.4.3 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.

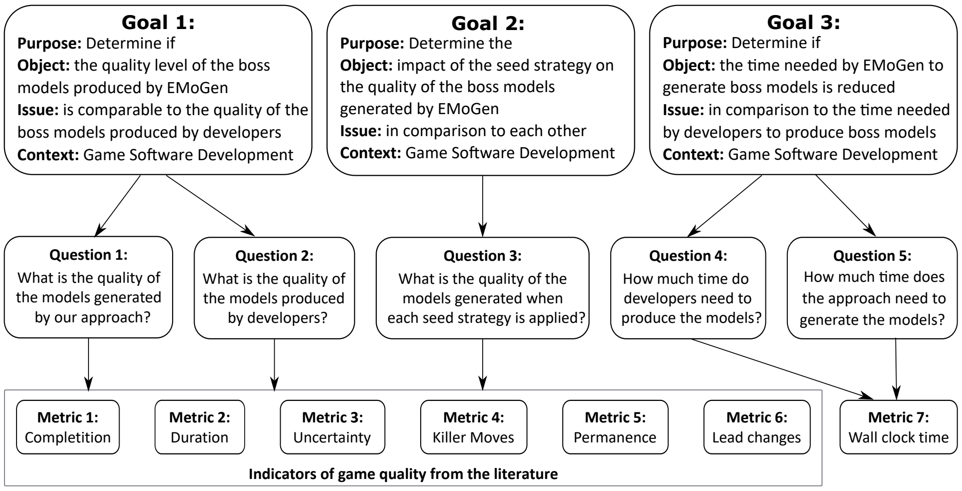

Goal 1:
Purpose: Determine if
Object: the quality level of the boss models produced by EMoGen
Issue: is comparable to the quality of the boss models produced by developers
Context: Game Software Development
Goal 2:
Purpose: Determine the
Object: impact of the seed strategy on the quality of the boss models generated by EMoGen
Issue: in comparison to each other
Context: Game Software Development
Goal 3:
Purpose: Determine if
Object: the time needed by EMoGen to generate boss models is reduced
Issue: in comparison to the time needed by developers to produce boss models
Context: Game Software Development
Question 1: What is the quality of the models generated by our approach?
Question 2: What is the quality of the models produced by developers?
Question 3: What is the quality of the models generated when each seed strategy is applied?
Question 4: How much time do developers need to produce the models?
Question 5: How much time does the approach need to generate the models?
Metric 1: Completition
Metric 2: Duration
Metric 3: Uncertainty
Metric 4: Killer Moves
Metric 5: Permanence
Metric 6: Lead changes
Metric 7: Wall clock time
Indicators of game quality from the literature

Figure 2.8: Application of the Goal Question Metric method for the evaluation

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.

2.5 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 [12] and Travassos [71]. They refer to in-silico, in-vivo, in-vitro, and in-virtuo. More recent works used models as experimentation units [57] within in-virtuo experiments [6], 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 [71].

We defined the experimental design of the evaluation following the Goal-Question-Metric (GQM) [12] 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. [13], we set three goals, which are defined through five questions that can be answered using seven metrics (see Figure 2.8). 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 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 [17]. 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.

2.5.1 Experimental Setup

Figure 2.9 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 2.9). 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.

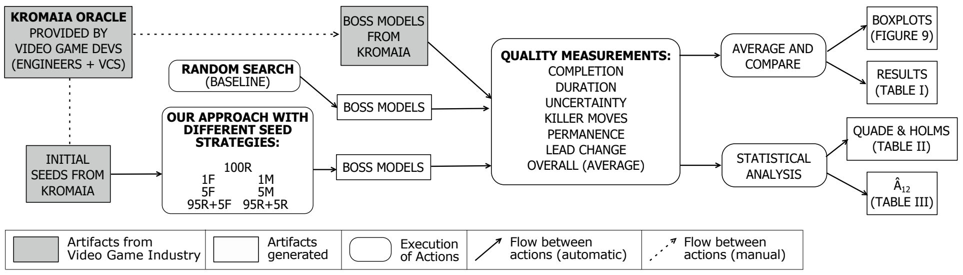


Figure 2.9: EMoGen Approach Evaluation Process.

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 [36].

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 [8], 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 [66, 2, 34, 16, 39, 26, 20, 15, 63] (see the middle part of Figure 2.9). 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.

2.5.2 Quality measurements

Previous research works have formalized fundamental and measurable indicators of game quality, like Depth and Decisiveness [66], Tension [39], Interestingness [2], Uncertainty [34], or Interaction [16]. 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 [17]. 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:

$$Q_{Completion} = \frac{Conclusions}{Duels} \tag{2.5}$$

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 ($T_{Optimal}$):

$$Q_{Duration} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels}\right) \tag{2.6}$$

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.

$$Q_{Uncertainty} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min(P_d, B_d)}{T_d}}{Duels}\right) \tag{2.7}$$

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%.

$$Q_{KMoves} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels}\right) \tag{2.8}$$

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:

$$Q_{Permanence} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d+K_d}}{Duels}\right) \tag{2.9}$$

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:

$$Q_{LChange} = clamp_{[0,1]}\left(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d+K_d}}{Duels}\right) \tag{2.10}$$

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:

$$Q_{Overall} = \frac{\sum_{i=1}^{N} Q_i}{N} \tag{2.11}$$

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.

2.5.3 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 [21, 4, 5]. 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`

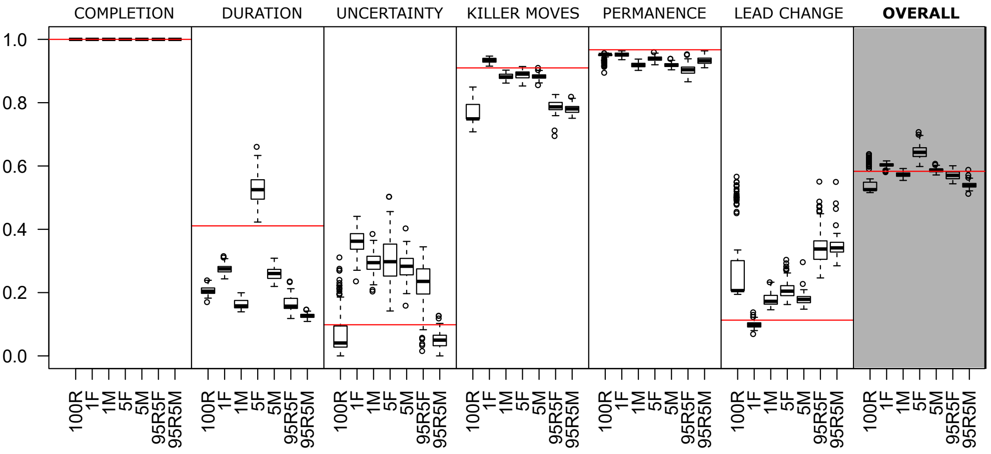


Figure 2.10: The results of the application of EMoGen with seven different seed strategies to generate final boss models. The results are grouped based on the 6 quality measures; the horizontal line in each group represents the value obtained by the original final boss models from Kromaia.

2.5.4 Results:

Figure 2.10 shows the results of the execution of our approach for each of the seven combinations of 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 2.9).

Each boxplot is generated from the results of 30 executions [8] 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 2.10 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 2.1 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 2.10 and in Table 2.1: 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 2.10, which cross each column 2.10, and the associated values from Table 2.1 (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.

Table 2.1: 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

2.5.5 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 [7]. The statistical analysis included a significance test, the corresponding post-hoc analysis, and an effect size measure.

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 [7]), we applied the Quade test since our data does not follow a normal distribution and the Quade test has proven [25] to be better than the rest of the non-parametric tests when working with real data.

The Quade test results in a $p\text{-}Value$ 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 [7], meaning that p-values below that number are statistically significant.

The results of the Quade test give a $p\text{-}Value$ 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.

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 [24]. 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 2.2 shows the results of the post-hoc analysis. Each column shows the $p\text{-}Value$ 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 2.2 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 2.2). It is common to obtain these results because the differences between some pairs of seed strategies are subtle.

Table 2.2: Holm's Post Hoc *p-Values* 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	$1.8x10^{-11}$

These results partially answer Q3. Taking into account the differences in the results from Table 2.1 and the fact that they are significant as shown by the Holm's post-hoc analysis (see Table 2.2), we can conclude that the selection of the seed strategy does have an impact on the quality of the final boss model produced in terms of the quality measurements included in this study.

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 [7]. 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 [29, 75]. $\hat{A}_{12}$ 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 2.3 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.

Table 2.3: 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 %

For instance, the fourth row (Table 2.3) 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).

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.

2.6 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 [50], 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 repair 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 [27], 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 [65] or Unreal, which uses its own DSL named BluePrints [23], are widely adopted by development teams, and their architecture is similar to that shown in Figure 2.4. 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.

2.7 Threats to validity

Following the guidelines suggested by De Oliveira et al. [19], 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 [17].

2.8 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).

Bibliography

[1] IGDA, International Game Developers Association, 2018 (cit. on p. 76).

[2] Ingo Althöfer. “Computer-Aided Game Inventing”. In: *Technical Report, Friedrich Schiller Universität Jena* (2003) (cit. on pp. 95, 96).

[3] Apostolos Ampatzoglou and Ioannis Stamelos. “Software engineering research for computer games: A systematic review”. In: *Information and Software Technology* 52.9 (2010), pp. 888–901. ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2010.05.004` (cit. on pp. 72, 78).

[4] Lorena Arcega, Jaime Font, and Carlos 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. DOI: `10.1145/3239372.3239392` (cit. on pp. 88, 99).

[5] Lorena Arcega et al. "An approach for bug localization in models using two levels: model and metamodel". In: *Software & Systems Modeling* (2019). ISSN: 1619-1374. DOI: `10.1007/s10270-019-00727-y` (cit. on pp. 88, 99).

[6] Lorena Arcega et al. "On the Influence of Models at Run-Time Traces in Dynamic Feature Location". In: *Modelling Foundations and Applications*. Ed. by Anthony Anjorin and Huáscar Espinoza. Cham: Springer International Publishing, 2017, pp. 90–105 (cit. on p. 92).

[7] Andrea Arcuri and Lionel Briand. "A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering". In: *Softw. Test. Verif. Reliab.* 24.3 (May 2014), pp. 219–250. ISSN: 0960-0833. DOI: `10.1002/stvr.1486` (cit. on pp. 102, 104).

[8] Andrea Arcuri and Gordon Fraser. "Parameter tuning or default values? An empirical investigation in search-based software engineering". In: *Empirical Software Engineering* 18.3 (2013), pp. 594–623. ISSN: 1573-7616. DOI: `10.1007/s10664-013-9249-9` (cit. on pp. 95, 100).

[9] Daniel 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 (cit. on p. 80).

[10] Daniel Ashlock, Colin Lee, and Cameron McGuinness. "Search-based procedural generation of maze-like levels". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 260–273 (cit. on p. 80).

[11] Daniel A Ashlock, Stephen P Gent, and Kenneth Mark Bryden. "Evolution of l-systems for compact virtual landscape generation". In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 3. IEEE. 2005, pp. 2760–2767 (cit. on p. 80).

[12] Victor R. Basili. "The Role of Experimentation in Software Engineering: Past, Current, and Future". In: Berlin, Germany, 1996, pp. 442–449. ISBN: 0-8186-7246-3 (cit. on p. 92).

[13] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. "The Goal Question Metric Approach". In: *Encyclopedia of Software Engineering*. John Wiley & Sons, 1994 (cit. on p. 92).

[14] James Bergstra and Yoshua Bengio. “Random Search for Hyper-parameter Optimization”. In: *J. Mach. Learn. Res.* 13.1 (Feb. 2012), pp. 281–305. ISSN: 1532-4435 (cit. on p. 75).

[15] George D. Birkhoff. *Aesthetic Measure*. Cambridge, Massachussetts: Harvard University Press, 1933. ISBN: 9780674734470 (cit. on p. 95).

[16] Cameron Browne. *Connection Games: Variations on a Theme*. Natick, Massachussetts: AK Peters, 2005. ISBN: 1568812248 (cit. on pp. 95, 96).

[17] Cameron Browne and Frédéric Maire. “Evolutionary Game Design”. In: *IEEE Trans. Comput. Intellig. and AI in Games* 2.1 (2010), pp. 1–16. DOI: `10.1109/TCIAIG.2010.2041928` (cit. on pp. 92, 96, 108).

[18] Cameron Bolitho Browne. “Automatic generation and evaluation of recombination games”. PhD thesis. Queensland University of Technology, 2008 (cit. on p. 80).

[19] Márcio De Oliveira Barros and Arilo Cláudio Dias-Neto. “0006/2011-Threats to Validity in Search-based Software Engineering Empirical Studies”. In: *RelaTe-DIA* 5.1 (2011) (cit. on p. 107).

[20] Havelock Ellis. *Impressions and Comments*. Boston, Massachussetts: Houghton Mifflin, 1914 (cit. on p. 95).

[21] Jaime Font et al. “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. Saint-malo, France: ACM, 2016, pp. 272–282. ISBN: 978-1-4503-4321-3. DOI: `10.1145/2976767.2976789` (cit. on pp. 88, 99).

[22] Miguel Frade, Francisco Fernandez de Vega, and Carlos 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 (cit. on p. 80).

[23] Epic Games. *Unreal Engine, Version 2018.3.9*. Cary, North Carolina, 1998 (cit. on pp. 72, 78, 107).

[24] Salvador García et al. "Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power". In: *Inf. Sci.* 180.10 (May 2010), pp. 2044–2064. ISSN: 0020-0255. DOI: `10.1016/j.ins.2009.12.010` (cit. on p. 103).

[25] Salvador García et al. "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". In: *Information Sciences* 180.10 (2010). Special Issue on Intelligent Distributed Information Systems, pp. 2044–2064. ISSN: 0020-0255. DOI: `https://doi.org/10.1016/j.ins.2009.12.010` (cit. on p. 102).

[26] Martin Gardner. "Theory of Everything". In: *The New Criterion* 23.2 (2004) (cit. on p. 95).

[27] David E. Goldberg. "Computer-aided gas pipeline operation using genetic algorithms". In: *Ph.D. dissertation* (1983) (cit. on p. 107).

[28] Heather J. Goldsby and Betty H. C. Cheng. "Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty". In: *Model Driven Engineering Languages and Systems*. Ed. by Krzysztof Czarnecki et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 568–583. ISBN: 978-3-540-87875-9 (cit. on p. 79).

[29] R. J. Grissom and J. J. Kim. *"Effect sizes for research: A broad practical approach*. Mahwah, NJ: Earlbaum, 2005 (cit. on p. 104).

[30] M. Harman and B.F. Jones. "Search-Based Software Engineering". In: *Information Software Technology* 43 (2001), pp. 833–839 (cit. on p. 75).

[31] Erin J Hastings, Ratan K Guha, and Kenneth 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 (cit. on p. 80).

[32] Erin J Hastings and Kenneth 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 (cit. on p. 80).

[33] Vincent Hom and Joe 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 (cit. on p. 80).

[34] Hiroyuki Iida et al. “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. DOI: `10.1007/978-3-540-28643-1\_41` (cit. on pp. 95, 96).

[35] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. “Polymorph: A model for dynamic level generation”. In: *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference.* 2010 (cit. on p. 80).

[36] David S Johnson. “A theoretician’s guide to the experimental analysis of algorithms”. In: *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges* 59 (2002), pp. 215–250 (cit. on p. 94).

[37] Stuart Kent. “Model Driven Engineering”. In: *Integrated Formal Methods.* Ed. by Michael Butler, Luigia Petre, and Kaisa Sere. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 286–298. ISBN: 978-3-540-47884-3 (cit. on p. 82).

[38] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press, 1992 (cit. on p. 76).

[39] Wolfgang Kramer. “What Makes a Game Good?” In: *The Games Journal* (2000) (cit. on pp. 95, 96).

[40] Levi HS Lelis, Willian MP Reis, and Ya’akov Gal. “Procedural Generation of Game Maps With Human-in-the-Loop Algorithms”. In: *IEEE Transactions on Games* 10.3 (2017), pp. 271–280 (cit. on p. 80).

[41] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. “Automatic track generation for high-end racing games using evolutionary computation”. In: *IEEE Transactions on computational intelligence and AI in games* 3.3 (2011), pp. 245–259 (cit. on p. 80).

[42] Andrew Martin et al. “Evolving 3d buildings for the prototype video game subversion”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2010, pp. 111–120 (cit. on p. 80).

[43] Michael McShaffry. *Game Coding Complete*. Paraglyph Publishing, 2003. ISBN: 1932111751 (cit. on p. 72).

[44] *Meta Object Facility (MOF) Version 2.4.1*. Object Management Group (OMG) Specification. 2013 (cit. on p. 82).

[45] Mariela Nogueira-Collazo, Carlos Cotta Porras, and Antonio J Fernández-Leiva. “Competitive algorithms for coevolving both game content and AI. A case study: Planet wars”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 8.4 (2015), pp. 325–337 (cit. on p. 80).

[46] Edward Rolando Núñez-Valdéz et al. “A model-driven approach to generate and deploy videogames on multiple platforms”. In: *J. Ambient Intelligence and Humanized Computing* 8.3 (2017), pp. 435–447. DOI: `10.1007/s12652-016-0404-1` (cit. on pp. 72, 78).

[47] Edward Rolando Núñez-Valdéz et al. “Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering”. In: *IJIMAI* 2.2 (2013), pp. 33–42. DOI: `10.9781/ijimai.2013.224` (cit. on pp. 72, 78).

[48] David Oranchak. “Evolutionary algorithm for generation of entertaining shinro logic puzzles”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2010, pp. 181–190 (cit. on p. 80).

[49] Chris Pedersen, Julian Togelius, and Georgios N Yannakakis. “Modeling player experience in super mario bros”. In: *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE. 2009, pp. 132–139 (cit. on p. 80).

[50] J. Petke et al. “Genetic Improvement of Software: A Comprehensive Survey”. In: *IEEE Transactions on Evolutionary Computation* 22.3 (2018), pp. 415–432. ISSN: 1089-778X. DOI: `10.1109/TEVC.2017.2693219` (cit. on p. 106).

[51] William L Raffe et al. “Integrated approach to personalized procedural map generation using evolutionary algorithms”. In: *IEEE Transactions on Com-*

putational Intelligence and AI in Games 7.2 (2014), pp. 139–155 (cit. on p. 80).

[52] Emanuel Montero Reyno and José Á Carsí Cubel. “Automatic Prototyping in Model-driven Game Development”. In: *Comput. Entertain.* 7.2 (June 2009), 29:1–29:9. ISSN: 1544-3574. DOI: 10.1145/1541895.1541909 (cit. on pp. 72, 78).

[53] Sebastian Risi et al. “Petalz: Search-based procedural content generation for the casual gamer”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 8.3 (2015), pp. 244–255 (cit. on p. 80).

[54] Jonathan Roberts and Ke Chen. “Learning-based procedural content generation”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 7.1 (2014), pp. 88–101 (cit. on p. 80).

[55] Christoph Salge and Tobias 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 (cit. on p. 80).

[56] B. Selic. “The pragmatics of model-driven development”. In: *IEEE Software* 20.5 (2003), pp. 19–25 (cit. on p. 72).

[57] Forrest Shull, Janice Singer, and Dag I. K. Sjberg. *Guide to Advanced Empirical Software Engineering.* 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 1849967121 (cit. on p. 92).

[58] Adam M Smith and Michael Mateas. “Answer set programming for procedural content generation: A design space approach”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 187–200 (cit. on p. 80).

[59] Adam M Smith and Michael 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 (cit. on p. 80).

[60] Jaime Solís-Martínez et al. “VGPM: Using Business Process Modeling for Videogame Modeling and Code Generation in Multiple Platforms”. In: *Computer Standards & Interfaces* 42 (2015), pp. 42–52. DOI: `10.1016/j.csi.2015.04.009` (cit. on pp. 72, 78).

[61] Nathan Sorenson and Philippe 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 (cit. on p. 80).

[62] Nathan Sorenson, Philippe Pasquier, and Steve DiPaola. “A generic approach to challenge modeling for the procedural creation of video game levels”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 229–244 (cit. on p. 80).

[63] George Stiny and James Gips. *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts.* Berkeley, California: University of California Press, 1978. ISBN: 0520034678 (cit. on p. 95).

[64] Adam Summerville et al. “Procedural content generation via machine learning (PCGML)”. In: *IEEE Transactions on Games* 10.3 (2018), pp. 257–270 (cit. on p. 80).

[65] Unity Technologies. *Unity, Version 2018.3.9.* San Francisco, California, 2005 (cit. on pp. 72, 78, 107).

[66] J. Mark Thompson. “Defining the Abstract”. In: *The Games Journal* (2000) (cit. on pp. 95, 96).

[67] Julian Togelius, Renzo De Nardi, and Simon M Lucas. “Making racing fun through player modeling and track evolution”. In: (2006) (cit. on p. 80).

[68] Julian Togelius, Mike Preuss, and Georgios N Yannakakis. “Towards multiobjective procedural map generation”. In: *Proceedings of the 2010 workshop on procedural content generation in games.* ACM. 2010, p. 3 (cit. on p. 80).

[69] Julian Togelius and Jurgen Schmidhuber. “An experiment in automatic game design”. In: *2008 IEEE Symposium On Computational Intelligence and Games.* IEEE. 2008, pp. 111–118 (cit. on p. 80).

[70] Julian Togelius et al. “Search-based procedural content generation: A taxonomy and survey”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186 (cit. on p. 80).

[71] Guilherme Horta Travassos and Márcio 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).* Roman Castles, Italy: IEEE Computer Society, 2003 (cit. on p. 92).

[72] Tim Tutenel et al. “Generating consistent buildings: a semantic approach for integrating procedural techniques”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 274–288 (cit. on p. 80).

[73] Muhammad Usman, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. “A product-line model-driven engineering approach for generating feature-based mobile applications”. In: *Journal of Systems and Software* 123 (2017), pp. 1–32. DOI: `10.1016/j.jss.2016.09.049` (cit. on pp. 72, 78).

[74] Roland Van Der Linden, Ricardo Lopes, and Rafael Bidarra. “Procedural generation of dungeons”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 6.1 (2013), pp. 78–89 (cit. on p. 80).

[75] András Vargha and Harold D. Delaney. “A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong”. In: *Journal of Educational and Behavioral Statistics* 25.2 (2000), pp. 101–132. DOI: `10.3102/10769986025002101`. eprint: `http://jeb.sagepub.com/content/25/2/101.full.pdf+html` (cit. on p. 104).

[76] James R. Williams et al. “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. DOI: `10.1007/978-3-642-23716-4\_13` (cit. on pp. 78, 79).

Chapter 3

Procedural Content Improvement of Game Bosses with an Evolutionary Algorithm

We present our Evolutionary Boss Improvement (EBI) approach, which receives partially complete bosses as input and generates fully equipped bosses that are complete. Additionally, the evolutionary algorithm and the new genetic operations included in EBI favor genetic improvement, which affects the initial partial content of the incomplete bosses originally provided. We evaluate our approach using Kromaia, a commercial video game released on PlayStation 4 and PC. EBI uses an evolutionary algorithm to evolve a population of bosses guided by duels between the bosses being generated and a simulated player. Our approach evaluates the quality, in terms of game experience, of both the bosses generated and those included in Kromaia using six metrics (Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change) from the literature. The results show that the quality of the bosses created by EBI is comparable to the quality of the original bosses that were manually created by the developers of Kromaia. However, the EBI approach reduces the time required to build the bosses from five months (of elapsed time as opposed to dedicated time) to just 100 minutes of unattended run. EBI enables developers to accelerate the creation of content, such as bosses, which is essential to ensure player engagement.

3.1 Introduction

Procedural Content Generation (PCG) is a topic that has become increasingly prominent within the Video Game community [57]. PCG is defined as the automated creation of video game content by means of algorithms [57].

Within the Software Engineering community, Genetic Programming (GP) [27] tackles the automated generation of software. GP and PCG share the idea of generation (software and video game content, respectively). In the last decade, the Software Engineering community has experienced a surge of interest in a subfield of GP known as Genetic Improvement (GI) [29]. GI improves existing software through an automated search, evolutionary algorithms, and search-based optimization. A recent survey [44] points out the main difference between GP and GI: GP builds a working program from scratch and GI uses an existing program as the starting point. Surveys on PCG [57, 21, 54] do not identify works that focus on addressing content improvement.

Because the establishment of GI as a subfield of GP has brought novel and positive contributions [29], the goal of our work is to simply recognize the content improvement work in PCG and give that subfield a name: Procedural Content Improvement (PCI). We consider that PCI might be as positive for the Video Game community as GI has been for the Software Engineering community.

In this work, we focus on game boss improvement. It consists in taking partially generated bosses as the starting point and generating complete bosses which are comparable or even better than the bosses completed by human developers in terms of game experience quality, i.e., how interesting the bosses are to players. Bosses are particularly powerful enemies that the player must overcome at the end of a stage or level. In order to help video game developers when they create game bosses, we present our Evolutionary Boss Improvement (EBI) approach. The EBI approach relies on an evolutionary algorithm that is guided by a simulated duel between the boss and the player.

We evaluate our approach using a commercial video game, Kromaia. This video game has been released worldwide in both physical and digital versions for PC and PlayStation 4. To create a boss, the Kromaia development team must perform the following steps: Creative Design, Spatial Organization, Behavior Specification, and Equipment Configuration.

When we apply our EBI approach to Kromaia, it first obtains a partially created boss enemy as input (i.e., Creative Design, Spatial Organization, and Behavior Specification have already been performed) and then performs the last step, Equipment Configuration. The output obtained from our approach is a complete boss.

In the evaluation, we compare the bosses generated by our approach with the five bosses that have been manually created by the development team of Kromaia, using six different indicators of the level of quality achieved by the game. These quality measures, which are taken from the video game research literature [12], are the following: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. The results show that the bosses generated by our approach outperformed those designed by the developers.

Additionally, the EBI approach generated those bosses after running unattended for 100 minutes, which is a significant advance in terms of time. The version control system used by the company that created the video game shows that the developers needed five months[1] to perform the last step (Equipment Configuration) before the final release of Kromaia.

Our approach helps developers to accelerate the creation of content. Content is often released as Downloadable Content (DLC), which is essential to reinforce player engagement and retention. Furthermore, content is also used to refresh existing products when they are released on new platforms.

The contribution of the present work could be summarized in this way:

- This work focuses on addressing the improvement of content, instead of software, and applies the ideas of PCG and GI to the production of

[1]This includes testing with real players.

complete content, taking as an input incomplete, partially generated content.

- In contrast with other previous works, our approach uses Evolutionary Computation instead of Machine Learning in order not to depend on knowledge bases, since such approaches may require datasets with thousands of examples [35], and those knowledge bases are not always available in the case of industrial products. Furthermore, even if there also exist previous works which generate content from a single sketch or draft of the complete content as a starting point, our work studies the improvement of a content which is received incomplete.
- Our work does not focus on game content such as levels, maps or sprites, which are typically represented by 2D images. In addition, the recent literature calls for works who address more diverse types of content, like characters and their skills. Our work focuses on the generation of final bosses, which are complex entities which are not internally represented as images, and belong to a type of content which has been less studied in comparison.

The paper is organized as follows. Section 3.2 summarizes related works. Section 3.3 provides the background. Section 3.4 presents an overview of our EBI approach. Section 3.5 presents the evaluation, comparing the results obtained by our approach with the content in the video game case study. Section 3.6 describes the threats to validity. Finally, Section 3.7 presents our conclusions.

3.2 Related Work

Several approaches focus on the generation of new content while others are more focused on the balance of existing content. Both types of approaches use similar methods for assessing and guiding the process [57] (direct fitness, simulations based on intelligent agents, or interactive evaluation with real users). Next, we describe the works belonging to these two categories.

3.2.1 Generation of new content

Our work leverages search to improve a specific type of non-playable character (NPC): the game boss. The previous surveys [57, 21] that cover search-based procedural content generation are about ten years old. To cover the years from these surveys, we ran a new search using the following query on Scopus: *TITLE-ABS-KEY ((pcg OR "automatic generation" OR procedural) AND (videogame OR game) AND ("search-based" OR evolutionary OR genetic OR "local search" OR "tabu search" OR "Monte Carlo Tree Search" OR mcts)).* The query returned 237 works. After manual inspection, we identified two works that tackle NPC and consequently are relevant to our work. We considered only those works that tackle NPC building, excluding works that are related to NPC but do not build NPCs (e.g., NPC placement [10] was excluded).

We classified the identified works following the distinction between Procedural Content Generation (PCG) and Procedural content Improvement (PCI) introduced in Section 3.1: PCG builds content from scratch and PCI uses existing content as the starting point. Two of the identified works belong to PCG as we show below.

Siqueira and Gadelha [47] tackle the generation of NPCs for massive multiplayer online real-time strategy games. They focus on generating heroes, which are the NPCs that manage soldiers in battle formations. Even though their approach takes as input a team of heroes for the defender side, it evolves the heroes of the attacker side. The latter are generated randomly according to a uniform distribution. In other words, the input heroes are not used as a starting point for creating heroes; the input heroes are used to assess the battle performance of the created heroes.

Ashley et al. [4] tackle the mating facet of NPCs in the context of artificial life. More specifically, they focus on a wolf-sheep predation model where they encode the mating partner preferences of each NPC. Their main idea is that a more effective mate selection increases the extinction time of the population. Their results show that NPCs evolve to favor mates who have survival traits. The implementation of their proposed approach in a video game remains as future work. In their work, NPCs are randomly initialized, that is, there is no other NPC used as a starting point.

In contrast, we tackle a different subtype of NPC (game boss) and we follow a PCI approach in which we take partially generated bosses as starting points. Our PCI approach enables developers to significantly accelerate the creation of NPCs (i.e., game bosses).

Even if a systematic literature review of PCI work is out of the scope of this study (actually, it is our future work), we also identify works that tackle the creation of shooter maps in the results of our search query. We focus on this type of content because the our video game case study is commonly described as a shooter game. Four of the identified works qualify as PCG, whereas one qualifies as PCI. Below, we present these works and why they belong to each category.

The work of Cardamone et al. [13] is the first to generate playable maps for an FPS (First Person Shooter). The authors propose four different representations for the levels and use the average fighting time of artificial agent simulations as the fitness function. Then, Lanzi et al. [30] go one step further by applying a similar approach to evolve shooter maps for match balancing. In other words, while Cardamone et al. aimed to evolve interesting maps (i.e., maps that allow the emergence of interesting game dynamics), Lanzi et al. focus on evolving a map that can improve the match balancing. Loiacono et al. [36][37] were the first to leverage multi-objective evolution for generating shooter maps. All of these works used the same encoding proposed by Cardamone et al. [13], the same static simulation through bots to guide the search, and the same case study (Cube 2). Furthermore, neither of these works reported that they use existing shooter maps as the starting point.

The work of Olsted et al. [64] also tackles the generation of shooter maps using the encoding by Cardamone et al. [13] and the Cube 2 case study. The novelty of this work is that it allows a group of human players to interactively and collectively evolve the shooter maps through voting. The human players guide the evolutionary search towards the map content that they prefer. The approach does not build shooter maps from scratch; their approach builds shooter maps from existing shooter maps that are considered to meet a minimum level of quality, thus preventing human players from receiving particularly bad maps.

It is worth mentioning that all of the above works on shooter maps use a significantly simplified version of Cube 2 maps. This might favor the use of PCG approaches, whereas tackling Cube 2 maps which are not simplified might benefit from PCI approaches. In the case of our work (which tackles game bosses instead of shooter maps), we did not simplify the content in any sense, and PCI enabled us to achieve a significant reduction in development time. Finally, in the manner of interactive approach of Olsted et al. [64], other interactive approaches for other types of content might also use PCI to avoid presenting content to human players if it is not good enough to be considered by them.

In our previous work [8], we tackled the PCG of game bosses of Kromaia. To do this, we leveraged the ideas of Model-Driven Engineering [25], which include the genetic operation of model repair and the use of the model interpreter to guide the evolution of bosses. In this work, we focus on improvement instead of generation from scratch. This work shows that improvement is beneficial in accelerating video game development. In addition, we achieve the positive results of improvement without the model repair operation and the model interpreter. This means that improvement can be used in video games for which the repair operation and the interpreter used as fitness are not available. Improvement allows developers to keep control of some steps of the creation process while they delegate other steps to automation (improvement). The developers of Kromaia acknowledge that some content requires that they keep some level of control, while other content may be completely generated from scratch. This suggests that PCG and PCI approaches can complement each other. In another previous work, we also used Kromaia as a case study for requirement traceability [7]. Traceability is important for video game developers since they are often asked to show how functionality has been implemented. For instance, Nintendo might ask developers to implement game saves in a specific way. Even though our previous work also uses Kromaia as a case study, the goals are completely different: traceability and improvement, respectively.

In addition and, in the context of PCG, there is an increasing interest in the application of Machine Learning (ML) and, specifically, Deep Learning (DL) to PCG in video games, as shown by a recent work which explores the evolution of this field of knowledge in the last years [34]. The paper surveys the techniques applied to content generation, and discusses both the limitations of the methods used and potential future research directions. For instance, this work exposes how systems which are autonomous, or merely receive initial parameters from human subjects, have difficulties to create quality content, and, therefore, mixed initiative generation [61], which include initial drafts or specifications given by humans for design assistance purposes [20], is gaining acceptance [34]. Such techniques are mainly used for the generation of platform game stages [19], and maps or 2D stages based on tiles [31][15] [34]. Our work uses an approach which does not receive a draft or sketch of the content generated, but incomplete content, resulting from previous design stages which are not directly related with the step that is necessary to complete the final bosses (Equipment Configuration).

The same survey shows how DL/ML works need datasets with thousands of examples [35], which is something that could be unavailable, for development teams who lack extensive knowledge databases. In the case of the company responsible for the video game case study, Kromaia, it was their first title and, even considering the possibility of generating content for a sequel, the original, finished video game includes few examples of final bosses. Additionally, companies often do not store the data necessary to create proper knowledge bases, a problem which was reported as knowledge vaporization [59], which is a reason for which, in our work, we use evolutionary computation instead of ML in order not to depend on knowledge bases.

The survey also discusses how some works managed to generate content as maps or sprites with DL approaches based on an single draft or sketch [34][63][32][50], or combining stages and levels from games [53][48][49]. However, in contrast with 2D sketches for final image content, our work studies the improvement of bosses, which are received incomplete and without a direct connection between such partial content, related to previous design stages, and the content pending. In addition, for the video game case study, the mix of content from different games is not possible, due to the lack of sequels and prequels, and the unique nature of the the bosses and the game at the time of its original release.

Finally, the survey explains that most of the works focus on the generation of content such as 2D game levels, stages or maps, and sprites, with such content being internally represented by 2D images [41]. Additionally, the article by Liu et al. calls for works who focus on more diverse types of content, such as characters in fighting games, and their skills and characteristics [34]. Our work is focused on the generation of less explored content in comparison with other types, according to the review: bosses, complex entities which are defined by means of a DSL and not represented as image content; these bosses behave as antagonist characters, and they are characterized by traits like their Equipment Configuration, which is the design stage studied by our approach.

3.2.2 Balance of content

Some approaches are designed as a companion for the development team in order to be used to gain insights about game balance by gathering information from simulation-based plays of artificial agents. In [23], the authors present an approach for balancing games through the use of restricted play agents. In [60], the authors make use of a multi-objective optimization algorithm to demonstrate the feasibility of an approach for automatic game balancing in the context of the Top Trumps card game. In [5], the authors propose a semi-automatic process for the game balancing of a prototype of a commercial video game (Zombie Village Game by Blue Byte GmbH). In [38], the authors create two different AI agents that are able to play a commercial board game (ticket to ride), generating information that can be used to balance the game.

These approaches are similar to the one presented in this work in the sense that they use artificial agents to gather information about the performance of the individuals being evolved. However, they do not focus on the creation of new content, and the resulting balanced content is not compared with content that has been balanced manually by developers, as our work does.

Other approaches gather the information for balance from the actions and knowledge of real users, and sometimes they adapt the content for those types of users. In [40], an evolutionary algorithm is given an initial population of pre-crafted and random Role-Playing Game (RPG) skills that are then assessed based on how often each skill is used by the players and evolved into new sets of skills. In [45], the approach focuses on informing game development about possible imbalances by means of agents that are trained using a dataset containing six months of plays of 213 human players of the game Aion, a Massive Multiplayer Online RPG. In [24], the authors use a deep-learning surrogate model to generate character classes of an FPS game that is tailored for a specific map. The approach uses a fitness function that takes into account the desired match duration and score and compares them to the values predicted by the surrogate model. The work in [6] applies an evolutionary strategy to generate playing card game decks by using a subset of the cards that are available in Hearthstone using artificial agent matches to evaluate the decks. Another work on the balance of the cards in Hearthstone [39] quantifies the impact of a change in a card on all of the sets of existing decks and game strategies. That work uses search strategies to determine which changes should be selected to balance the game.

These approaches benefit from the knowledge of several users, which can be used to guide the balancing process and even tailor the results for specific users or styles of play. However, in our work, we only use an artificial agent to guide the process and then compare the resulting content to the (supposedly good) content created originally by the developers. Nevertheless, tailoring the process of boss generation to specific types of players or play styles is something that we will explore in the future (e.g., in Kromaia, since there are different types of ships controlled by the player in the game, suggesting that different archetypes of players are playing the game).

3.3 Background

In this work, we focus on bosses, which are created in different steps using a Domain-Specific Language (DSL). This section presents the creation steps and the DSL.

3.3.1 Steps for Creating Bosses

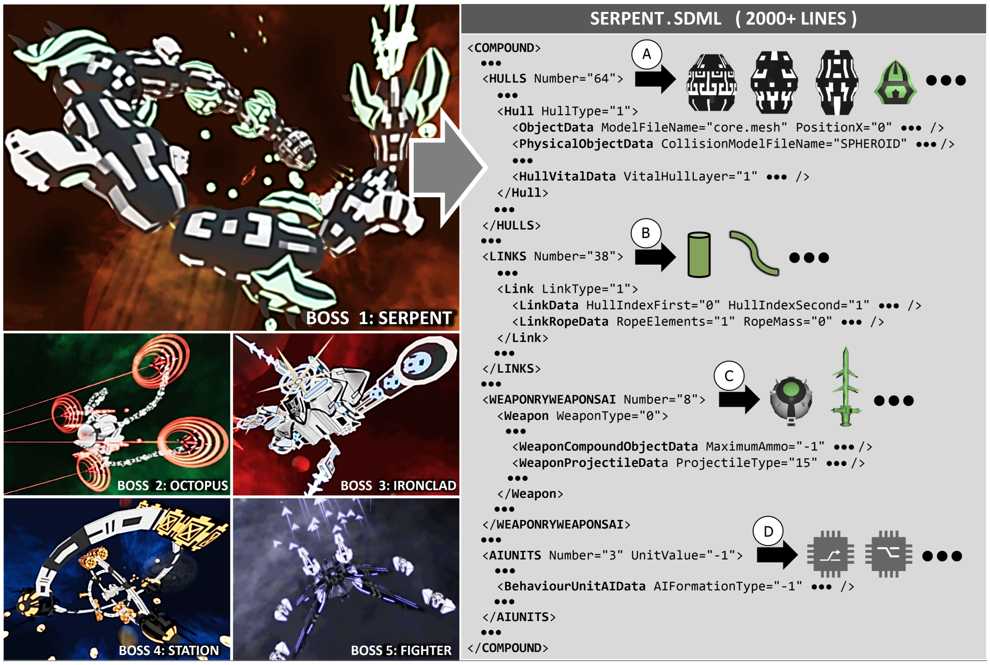
The creation of bosses in video games involves several steps. In our video game case study (Kromaia), the development team followed a four-step process. First, in the creative design step, a general specification of the bosses is provided, determining its structure, anatomical constraints, and visual appearance. Then, in the spatial organization step, the anatomical specification of the bosses is performed, arranging a set of hulls into a specific disposition and specifying relationships and hierarchies among the hulls. The third step is the behavior specification, which determines the artificial intelligence that will drive the boss.

The final step is the Equipment Configuration, which deals with weak point and weapon configuration. Determining the presence of different attack/defense items and the hulls that they are attached to has a significant impact on both the difficulty associated with the unit and the user's experience. This delimits the power assigned to the boss as well as the valid strategies that the players can use to defeat that boss.

In this work, we focus on the Equipment Configuration step to improve the bosses of Kromaia, the video game case study. This step does not refer to the task of tuning parameters that represent modifications of fixed content: it refers to the addition of procedurally generated content that is classified as Necessary in PCG taxonomy research works [57, 62]. This is because, before the Equipment Configuration step, the bosses lack weapons and weaknesses, and, therefore, they are not suitable for the game.

3.3.2 Domain-Specific Language for Shooters

This section presents SDML (Shooter Definition Model Language), which is a DSL created by Kraken Empire, the company responsible for the development of the video game Kromaia. SDML covers the definition of every mission, vehicle, creature, and landscape of the game. The left part of Fig. 3.1 shows different bosses that are included in the video game case study; the right section of Fig. 3.1 shows part of the SDML content in `Boss 1: Serpent`. The main SDML concepts that are relevant to the bosses are:



```
<COMPOUND>
  •••
  <HULLS Number="64">
    •••
    <Hull HullType="1">
      <ObjectData ModelFileName="core.mesh" PositionX="0" ••• />
      <PhysicalObjectData CollisionModelFileName="SPHEROID" ••• />
      •••
      <HullVitalData VitalHullLayer="1" ••• />
    </Hull>
    •••
  </HULLS>
  •••
  <LINKS Number="38">
    •••
    <Link LinkType="1">
      <LinkData HullIndexFirst="0" HullIndexSecond="1" ••• />
      <LinkRopeData RopeElements="1" RopeMass="0" ••• />
    </Link>
    •••
  </LINKS>
  •••
  <WEAPONRYWEAPONSAI Number="8">
    <Weapon WeaponType="0">
     •••
      <WeaponCompoundObjectData MaximumAmmo="-1" ••• />
      <WeaponProjectileData ProjectileType="15" ••• />
     •••
    </Weapon>
    •••
  </WEAPONRYWEAPONSAI>
  •••
  <AIUNITS Number="3" UnitValue="-1">
    •••
    <BehaviourUnitAIData AIFormationType="-1" ••• />
    •••
  </AIUNITS>
  •••
</COMPOUND>
```

Figure 3.1: Bosses in Kromaia, defined with SDML. The content included in this commercial video game (e.g., vehicles, creatures, worlds, and missions) is described in SDML. These include a wide range of data covering geometry, physics, contraptions, and AI, and follow a modular structure.

- **Hulls and Links:** The Hull Module is a collection of solid bodies that are connected through configurable joints or links. This hierarchy defines the anatomy and physics of the boss. Depending on the arrangement and the flexibility of the links used, bosses may have rigid structures, mobile parts, or even segmented tentacles. Fig.3.1 shows how visual appearance, geometry, and physics-related attributes are defined for hulls (see A) and the varied link types (see B).

- **Weak Points:** These are damageable objects that are attached to certain hulls. They can be organized in layers that are progressively unlocked as an enemy player destroys them. Any opponent trying to defeat a boss must first destroy these weak points. In the beginning, they are the only damageable objects. However, once every weak point has disappeared, the normal hulls become damageable, and, therefore, it is possible to destroy the boss. Fig. 3.1 includes an example of a weak point (marked as `HullVitalData`) that belongs to a specific layer.

- **Weapons:** These are objects that are capable of inflicting damage by using bullets, launching homing missiles, tracing rays, or affecting the target on direct contact. The bosses included in the commercial releases of Kromaia use these four weapon types. These weapons involve AI automatic gun turret behaviors that make them aim at targets (players). An example of an AI weaponry module and possible parameters for a weapon is shown in Fig. 3.1(C).

- **AI components:** These concepts define the way that bosses act during a game in terms of artificial intelligence. An AI module can include several AI components for different situations, as shown in Fig. 3.1(D), and they also can describe flocking behaviors.

In the video game case study, an average boss unit has 64 hulls. Each hull might have, simultaneously, one of the four possible weapon types and a weak point. This is a design feature that was included by the developers, not a decision made during this research study. This turns out to be $2^{64*(4+1)} = 2^{320}$ possibilities for equipment configuration, which shows that testing every single possibility is not feasible. Our EBI approach explores this search space using an evolutionary algorithm.

3.4 Overview of our EBI Approach

This section presents our EBI approach, the goal of which is to improve bosses. It receives a partially configured boss as input that is used to encode the population of an evolutionary algorithm. Then, the individuals are assessed by a fitness function and evolved through the Improvement Crossover and Improvement Mutation operations. This is repeated until the stop condition is met. Finally, EBI decodes the best individual generated into a completely configured boss.

In the following subsections, we present the evolutionary algorithm, the encoding, the genetic operations, and the fitness for the EBI approach.

3.4.1 EBI Algorithm Summary

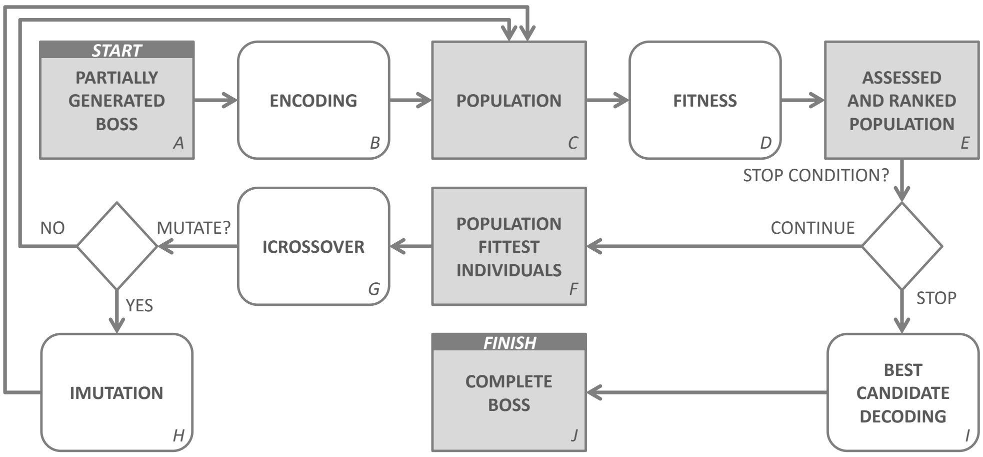


Figure 3.2: EBI Approach Overview.

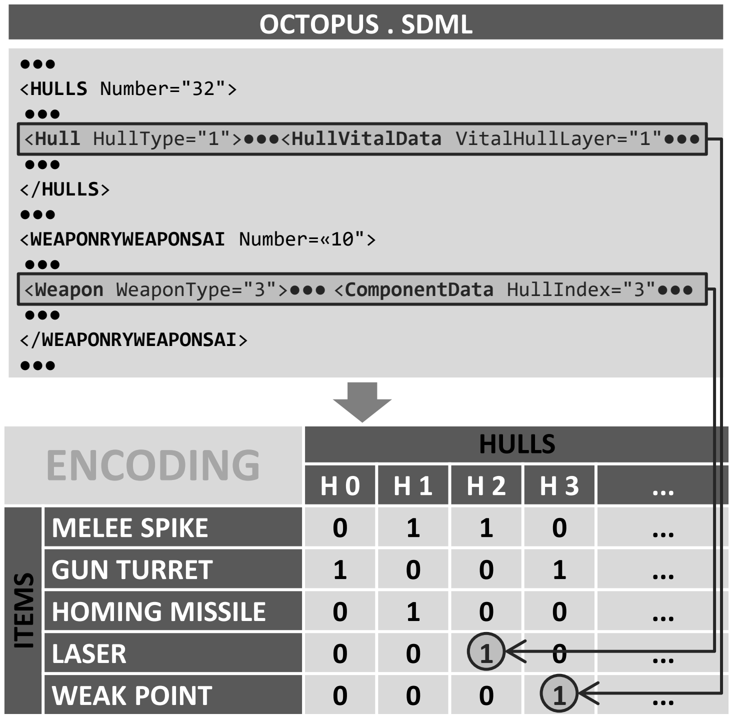
The evolutionary algorithm used by EBI iterates an Equipment Configuration configuration population and makes that population evolve by means of our genetic operations. A fitness operation guides the evolutionary algorithm, which tries to maximize the fitness values of the individuals included in the population. The fittest individuals reproduce and the population size is controlled and remains stable by discarding the lesser promising configurations. Fig. 3.2 shows the main steps included in our EBI approach.

- Input - Partially Configured Boss (Fig. 3.2, A): This is a boss, defined by means of SDML that has gone through three of the four creation steps (Creative Design, Spatial Organization and Behavior Specification).
- The Encoding of the possible realizations of the Equipment Configuration (Fig. 3.2, B) is necessary before creating the initial population which will be evolved by the evolutionary algorithm (Fig. 3.2, C). These encoded, randomly generated configurations represent the last step in the boss creation process.
- The Fitness step (Fig. 3.2, D) evaluates the Equipment Configurations by assigning values to them and sorting the population as a ranking (Fig. 3.2, E). These fitness values depend on the results obtained from the simulation of duels between a human player and the bosses included in the population, which use their Equipment Configurations. The process is over when an Equipment Configuration with a fitness value that is high enough is found or when a time limit is reached.
- Assuming that the process is not over yet, the next step uses improvement focused genetic operations to create a new generation of Equipment Configurations (Fig. 3.2, G, H). Those configurations with the highest fitness values in the population are selected and allowed to reproduce by means of pairing (Fig. 3.2, F). Then, new Equipment Configurations are obtained crossing the genetic material of the possible combinations of two potential parents from the selected group. Finally, this step also introduces changes in the new configurations using Mutation, an operation that could make the new Equipment Configurations surpass the fitness values achieved by their parents or get worse than them.
- Output - Completely Configured Boss: This is a complete boss (including the Equipment Configuration) which has been decoded back to SDML (Fig. 3.2, I, J).

3.4.2 Boss Equipment Configuration Encoding of the EBI Approach

The Equipment Configurations obtained by our approach are the candidates that are generated for the last configuration stage in the boss creation process. Nevertheless, these configurations must be encoded. In evolutionary algorithms, this is usually done by representing candidates as binary strings or arrays containing values such as true/false or 0/1.

In the EBI approach, the encoding for the Equipment Configurations is as follows: each Equipment Configuration is a binary, bi-dimensional matrix in which columns correspond to the hull collection for the boss studied, and each row represents weak points and the four weapon types used by the bosses. The values in each cell indicate the presence (1) or absence (0) of a certain item type in a hull. The encoding used to represent Equipment Configurations is shown in Fig. 3.3.


OCTOPUS . SDML

```
●●●
<HULLS Number="32">
 ●●●
<Hull HullType="1">●●●<HullVitalData VitalHullLayer="1"●●●
 ●●●
</HULLS>
●●●
<WEAPONRYWEAPONSAI Number=«10">
 ●●●
<Weapon WeaponType="3">●●● <ComponentData HullIndex="3"●●●
 ●●●
</WEAPONRYWEAPONSAI>
●●●
```

ENCODING		HULLS				
		H 0	H 1	H 2	H 3	...
ITEMS	MELEE SPIKE	0	1	1	0	...
	GUN TURRET	1	0	0	1	...
	HOMING MISSILE	0	1	0	0	...
	LASER	0	0	1	0	...
	WEAK POINT	0	0	0	1	...

Figure 3.3: Example showing the relation between data present in SDML and Equipment Configuration encoding.

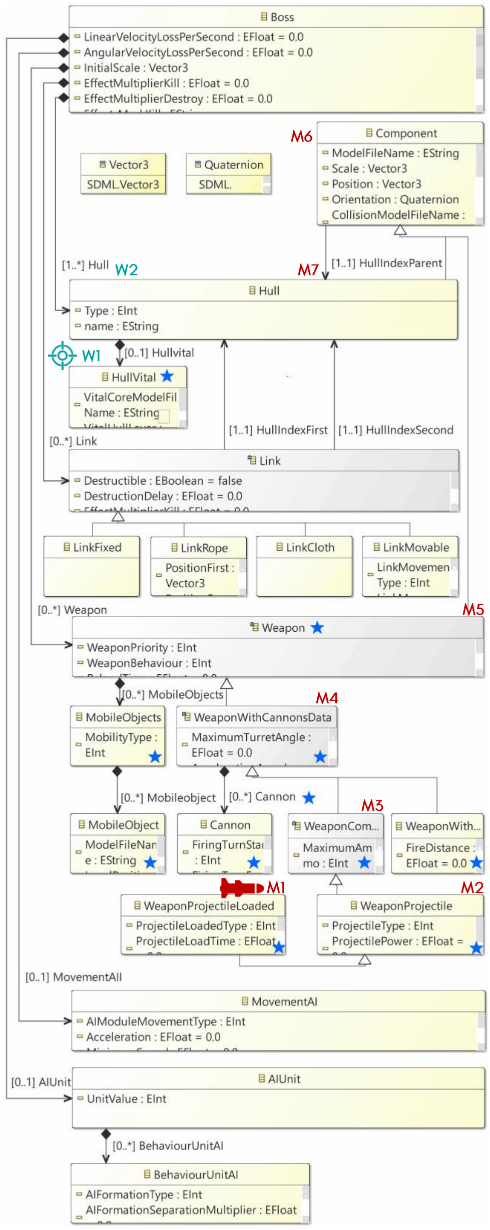


Figure 3.4: Fragment of the metamodel which describes the rules for creating bosses in Kromaia. The elements marked with stars indicate the parts of such excerpt which are involved in the Equipment Configuration step. The series of marks M1-M7 and W1-W2 are examples which show the elements involved in terms of support for missiles (a type of weapon) and weak points, respectively, beyond Equipment Configuration.

3.4.3 Genetic Operations of the EBI Approach

Our work focuses on improvement, and this means that our approach deals with content which has been partially generated previously. In such circumstances, manipulating content by means of evolutionary computation and, therefore, using operations which mix and modify individuals of a given population, imply considerations beyond the changes or additions provided by traditional genetic operations.

Taking into consideration that the complete creation process of a specific content would involve a set of elements, properties, and rules which connect them, our work deals with improvement as the addition, removal, or modification of specific fragments of such set. However, if such modifications only consider those specific fragments, the new or updated content could be invalid and, therefore, not usable, due to the fact that the fragments are not isolated and independent.

In the context of the case study with which our approach is applied, an example of the aforementioned issue would be weak point designation within final bosses. Fig. 3.3 shows that the encoding used by our approach, which represents elements and properties corresponding to the last step, Equipment Configuration, includes information relative to weak points. In terms of properties and internal definition, weak points are different from the rest of hulls or solid bodies, but the inclusion of hulls in the structure of a final boss corresponds with one of the first stages (Spatial Organization). Therefore, the changes introduced during Equipment Configuration are not independent and isolated.

Recent works have surveyed a compendium of Search-Based approaches, which were developed in the last decades and made use of crossover and mutation operations [43][9]. These works show how methods like single-point crossover and random mutation are the dominant choices. In addition, the literature describes how, within the community of general evolutionary computation research, there exists a significantly high number of different genetic operations, more than 20 and 50 mutation and crossover operations, respectively [33][42][42]. However, the recent surveys do not describe genetic operators which are designed in order to focus on improvement and its implications. Instead of that, the scope of the operators surveyed is not a specific part of the individuals manipulated, and they cross or mutate complete solution candidates.

In this work, we propose new genetic operations which take into account the particularities implied by the notion of improvement. In order to define such improvement-focused operations, we take into account Model-Driven Engineering (MDE) [26] and consider the abstract representation of knowledge, like the content improved by our approach. More specifically, in the context of MDE a metamodel represents the formalization of the characteristics and particularities of the models which will be created in accordance to it [26].The metamodel expresses the relationship between the different elements involved in the definition of an entity, the nature, and cardinality (if applicable) of such relationships, and the properties included. In the case of video games, the content may be formalized, for instance, by means of tools like Blueprints, in the case of the commercial engine Unreal [17], or the DSL used by the developers in the video game case study of this work. Models may formalize content such as objects, characters, or behaviors [46][55]. Our new operations make use of metamodels in order to focus on the fragments which complete the partially generated content and the binary encoding of such content. The metamodel is also used in order to modify, if it is necessary, the fragment of the partially complete content originally provided, which is not encoded, to keep the complete content coherent and valid.

In Kromaia, the video game case study, it is necessary to take into account and use certain rules and constraints which must be observed in order to create a suitable final boss. Every step in the production of a boss, in addition to Equipment Configuration, must be completed in accordance to those specifications. The compliance of the bosses generated with them is relevant with regard to the idea of improvement used in our work, and how such improvement affects a complete boss, even if our EBI approach focuses on one of the steps involved in the creation of a boss. This is relevant to the application of the genetic operations of our approach, since the concepts managed and added to the bosses by EBI, which are related to the Equipment Configuration step, could imply changes which affect the concepts corresponding the previous steps. Our approach takes advantage on the fact that the developers of the video game case study used SDML not only to define bosses, but also to formalize the characteristics which the bosses themselves should met in order to be considered as valid. Such formalization is shown in Fig. 3.4, which includes a fragment of the metamodel used to define the bosses included in the video game case study. This metamodel is used by our approach to correctly conduct improvement, which is driven by the modifications done by the evolutionary algorithm and our genetic operations, which focus on Equipment Configuration.

Our EBI approach generates new Equipment Configurations using some of the existing ones as parents. This process is supported by the new operators which we propose in this work: ICrossover (Improvement Crossover) and IMutation (Improvement Mutation). These genetic operations are based on crossover and mutation, two operators which are widely present in the evolutionary computation research literature [16] and they are used in order to expand populations and introduce variability into them. We created operators based on them in order to work in favor of improvement in the context of our work. The genetic operations are used to add diversity to the population and, eventually, to lead to individuals which are fitter than their ancestors.

First, the operators are adjusted to work with Equipment Configurations which represent possible weak point and weapon distributions in a boss. This configuration is the last step involved in the creation of a complete boss, while the previous steps provide the partially generated bosses to which our approach applies an EA.

The fittest Equipment Configurations within the population are selected as parents for the new offspring. The fitness value calculation process is described in Subsection 3.4.4. Once the population is sorted, the best 10% of the Equipment Configurations are selected and pairwise combined to generate the new offspring. Each pair of parents is used to generate a new Equipment Configuration by applying the ICrossover operator. The newly created Equipment Configuration could be randomly designated to undergo mutation, by means of the IMutation operator. An elitism of 55% is applied, so the best individuals remain unchanged for the next generation.

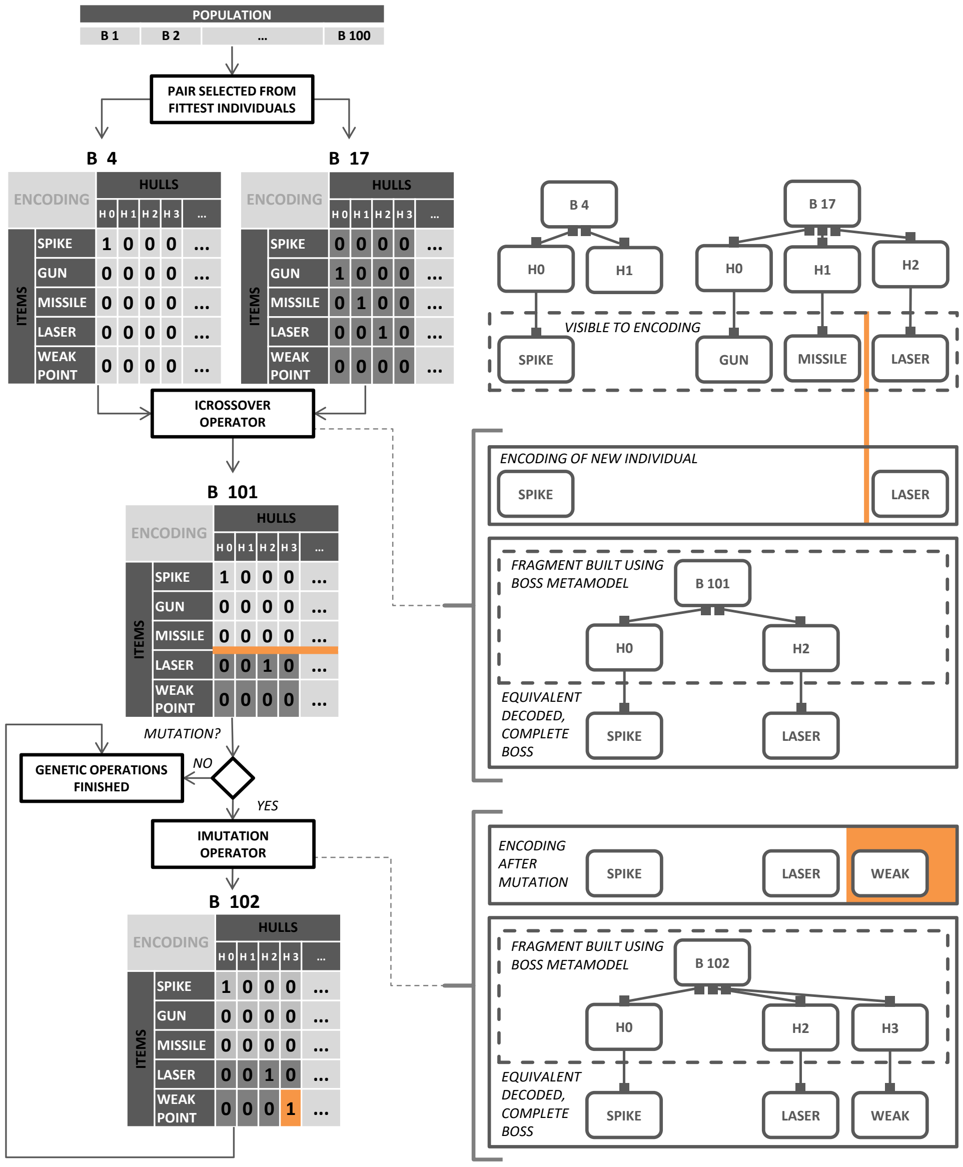


Figure 3.5: Genetic Operations overview. The examples show how improvement is driven by the genetic operations, which give priority to Equipment Configuration but influence the partial completion fragment originally provided, taking advantage on the metamodel.

- **ICrossover:** The ICrossover operation mixes the content of two individuals to create a third, new one. Like the methods used traditionally in evolutionary computation, it is used to increase a population through the combination of the genetic material of two existing individuals. In the context of our approach and the video game case study, the encoding is a bi-dimensional matrix, but the crossover operation processes an Equipment Configuration as if it were a one-dimension array that contains a consecutive collection of rows of binary elements. Therefore, the first element corresponds to the value placed in the first row and the first column, and the last element in the configuration refers to the value present in the last column and the last row. The improvement crossover operation used by our approach gives priority to the concepts introduced by the last step, Equipment Configuration. Those concepts reflect the part of the metamodel related to the use of weak points and weaponry, as shown by the sections highlighted and marked with stars the fragment of the metamodel included in Fig. 3.4. These are the steps necessary to perform the ICrossover operation:
  - First, the genetic material of both parents is combined following the method known as single-point crossover. Assuming that the encodings of the two parents have the same size, a random position is selected and it marks a reference point for the new individual: the genetic material within the region previous to such point is taken from the first parent, while the rest is inherited from the second parent. The resulting individual could eventually be superior to its progenitors.

    In the case of the application of our approach to Kromaia, the weaponry and weak point elements, which are represented in the binary encoding and correspond with the Equipment Configuration step for both parents, are combined. First, a random value n in the interval [0, S-1] is selected (S is the size of the configuration, interpreted as a one-dimension array). Then, since every Equipment Configuration for a certain boss has the same size S, the new configuration takes its first n array elements from the first parent and the last S - n elements from the second parent. Depending on the fitness value given to a new configuration, it could outperform its parents or be selected among the best in the population to generate new configurations in a later iteration of the evolutionary algorithm.

The top right part of Fig. 3.5 shows how the encoded configurations for both the parents and the new individual produced by means of ICrossover only contain information relative to the Equipment Configuration step.

– Once the new individual has been created, the impact of the mixed genetic material encoded is studied. This study is done in order to determine possible changes required with regard to the fragment not encoded (but included in the partially completed content which is provided to our approach) of the individual. Such modifications done on decoding are necessary to keep the complete content valid and usable. Therefore, even if the operation is focused on the fragment which would complete the individual, it implicitly drives the improvement since it could lead to necessary modifications on the partially completed fragment of the content which is received as an input by our approach.

  In the case of the video game case study, once the genetic material corresponding to the Equipment Configuration step has been mixed, the impact of the elements for such step is calculated making use of the metamodel. The elements included in the encoding for the new individual created determine what other elements, which are necessary to make the complete boss feasible and are not explicitly present in the encoding, are required too. Those elements can be taken from the parents indirectly, thank to the information available in the metamodel. Therefore, the improvement process, which can involve the elements corresponding to different steps of the creation of a boss, is actually driven by the last step, the Equipment Configuration, since the encoding is focused on the elements related with that step. The center right part of Fig. 3.5 shows how the improvement process could lead to an improved boss in terms of other aspects, like internal structure.

- **IMutation:** The improvement mutation operation is inspired by the mutations found in biology. These mutations make modifications in the genes of individuals that are caused by random factors. The IMutation operation is first focused on producing modifications related to the completion of content which is provided partially. In the context our EBI approach and the video game case study, and similarly to the ICrossover operation, this means that IMutation gives priority to the elements corresponding to the Equipment Configuration step:
  - First, the operation traverses all the elements which belong to the encoding and determines if each of them is mutated according to a certain probability.

    In the case of the bosses from Kromaia a bit string mutation is applied on the new Equipment Configurations, which are created through ICrossover operations.
  - The resulting individual may include changes that would make it not usable or valid unless the part not encoded is redefined.

    With regard to the bosses for Kromaia, the metamodel is used in order to identify the elements which are not included in the encoding and have been indirectly affected by the mutation. The bottom right part of Fig. 3.5 shows that, for instance, adding weak points could imply structural changes which are not taken into account in the Equipment Configuration step. The right part of Fig. 3.5 provides a general overview of the implications of crossing and mutating for partially generated bosses in Kromaia. In addition, Fig. 3.4 shows how, for example, the ability to launch missiles (Mark series M1-M7), or adding a weak point (W1-W2) implies the presence of certain elements which must be included in order to support those characteristics.

3.4.4 Fitness of the EBI Approach

The fitness step in our EBI approach determines the value of each Equipment Configuration that is generated. This stage takes an Equipment Configuration population as input and then measures the suitability of each configuration for the problem studied. Once this step is finished, it generates a ranking that sorts the Equipment Configurations according to the fitness values obtained so that the configuration with the best fitness value is ranked first. The evolutionary algorithm in our approach uses this fitness value to select parents for the next generation of configurations and to obtain the highest ranked Equipment Configuration once the search is over.

To obtain the fitness value for an Equipment Configuration, the EBI approach simulates a duel between a boss that uses that configuration and a player. It is an unattended duel in which both contenders are simulated. In that simulation, the player visits the different regions in the boss and tries to destroy the weak points that are available, while the boss uses the weapons in that configuration, which are different in power and range, to try to defeat the player. The simulation uses the information considered to be relevant by the developers to perform a simulated confrontation and includes statistical values regarding weapon accuracy, damage probability, and average player precision. The duel simulation is not deterministic, and it uses player and boss AI simulated agents that are handled in terms of actions, attacks, and damage by a state machine-based system. State machines have been used to describe boss behaviors in previous works [52]. The player agent performs a cyclic itinerary that successively focuses on each of the hulls included in the boss. When the player agent is focused on a hull that contains weak points, it tries to destroy them. The boss agent attacks the player using the weapons present in the different hulls. The weapons that are present in the hull on which the player is currently focused have a higher probability of hitting the player, which decreases with distance. The inherent accuracy and probability of success for the different types of weapons vary as well: Melee weapons that are present in distant hulls are not effective, but distant lasers or missiles have a reduced, yet non-zero probability of success. Both the player and the boss actively try to win the match and do not run away. This avoids draws and ensures that one of the contenders wins. The objective of the player is to destroy the weak points included in the boss, while the attacks performed by the boss aim for the player unit as a whole. Since the duel is not deterministic, it is run 30 times, following the suggestion in [3]. Once the simulation is over, our approach has enough information on relevant events and the progress of the duel to calculate a fitness value.

The developers provided two main measures that the bosses in the video game should maximize in order to prove their suitability for commercial releases: the player victory percentage ($F_{Victory}$), and the player optimal health percentage after a victory (F_{Health}). The fitness measures in this work (and the quality measures described in Section 3.5) use the following function:

$$\boxed{clamp_{[0,1]}(x) = max(0, min(x, 1)) \qquad (3.1)}$$

Our approach calculates the $F_{Victory}$ criterion as a measure of the difference between the number of victories achieved by the player (V_P) and the desired, optimal number of victories ($V_{Optimal}$) (33%, according to the criteria used by the developers):

$$\boxed{F_{Victory} = clamp_{[0,1]}\Big(1 - \frac{|V_{Optimal} - V_P|}{V_{Optimal}}\Big) \qquad (3.2)}$$

The F_{Health} criterion (for duels won by the player) is the average difference between the health percentage of the player at the end of the duel (Θ_P) and the optimal life level that the player should ideally keep at that point ($\Theta_{Optimal}$, 20%, according to the developers). This criterion focuses on how adequate the victories achieved by the player are, and it considers the health level of the player after defeating a boss in those duels:

$$\boxed{F_{Health} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{V_P} \frac{|\Theta_{Optimal} - \Theta_P|}{\Theta_{Optimal}}}{V_P}\Big) \qquad (3.3)}$$

$F_{Overall}$ is a direct measure (as intended by the developers) that calculates an average fitness value for an Equipment Configuration, including every specific fitness criterion studied:

$$\boxed{F_{Overall} = \frac{F_{Victory} + F_{Health}}{2} \qquad (3.4)}$$

In the end, $F_{Overall}$ is a value in the interval [0, 1] that allows the EBI approach to create an Equipment Configuration ranking.

3.4.5 Situating the approach

The present work applies an EA to partially completed game bosses in order to automatically produce complete bosses that have equal or better quality (in terms of six metrics) than the bosses that were originally completed by the developers. According to available taxonomies for procedural content generation [57, 21, 51, 62], our work can be classified as follows. The content type generated is the equipment configuration of boss enemies, and the method of PCG used is an Evolutionary Algorithm that manipulates SDML (Shooter Definition Model Language) models. With regard to the nature of the content, the multiplicity is single instance, since the case study is a single player game; the content generated is necessary and the type of derivation is built-in (since the elements generated are part of the game). With regard to the generation process, the mode is offline since it is performed as part of the development of the game. The degree of parameterization can be classified as parameter vectors (the input provided is a partially built boss). The nature of the process is stochastic and the EA follows a 'generate and test' constructiveness. The generation is non-personalized for different players; the generation is not controlled by the player. Finally, with regard to the game dependence, Kromaia belongs to the 3D space shooter genre, within the entertainment industry of commercial video games.

3.5 Evaluation

This section presents the evaluation of our approach: the research questions, the quality measurement, the experimental setup, the implementation details, and the results obtained.

3.5.1 Research Questions

The following questions address the evaluation of our EBI approach taking into account its results and the time necessary to obtain them:

RQ$_1$: *Does our EBI approach provide completely configured bosses that are comparable (or even better) in quality to those designed by the developers of the video game case study?*

RQ$_2$: *Does our EBI approach reduce the time necessary to configure good quality bosses in the video game case study?*

3.5.2 Quality Measures for the Configurations

The bosses included in the commercial release of the video game case study use Equipment Configurations that were approved after studying, adjusting, or even discarding several alternatives. The development company provided information regarding the version control system to measure the time invested in the bosses. The revision log showed that, after completing the Creative Design, Spatial Organization, and Behavior Specification steps, the Equipment Configuration step for each of the bosses required a month before the results were satisfactory. Thanks to the feedback given by the players, the developers determined that the users liked these boss units and found them to be enjoyable.

In order to compare the results obtained by the developers and our approach, we use criteria that measure the quality of the bosses. As previous works describe, in the context of video game development, quality refers to the probability of a game experience being considered interesting by users [12] in terms of intellectually challenging content. These works also state that, in general, players can express whether or not they consider a game experience to be enjoyable; however, they usually find it difficult to express the reasons precisely.

There are measurable indicators that have been studied in the past and are considered to be relevant. Tension [28], Decisiveness [56], Interestingness [1], Interaction [11], and Uncertainty [22] were described in previous research works as being fundamental game quality indicators. More recently, Browne et al. showed through experiments with users that there is a set of criteria that stand out and are the most important: Lead Change, Permanence, Killer Moves, Uncertainty, Duration, and Completion [12]. The evaluation included in our approach gives each of these criteria a value in the interval [0, 1]. In our approach, the quality measures are calculated using data that is retrieved from a simulation, which involves 30 non-deterministic confrontations (*Duels*). For the video game case study, the developers determined through tests and questionnaires with players 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 $2 * T_{Optimal}$ (20 minutes).

- **Completion (Viability):** The criterion $Q_{Completion}$ calculates a ratio of conclusions over total duel count:

$$Q_{Completion} = \frac{Conclusions}{Duels} \tag{3.5}$$

$$\begin{aligned} Duels &= Total\ number\ of\ duels \\ Conclusions &= Number\ of\ duels\ won\ by\ either\ side \end{aligned}$$

  A game against a boss unit should end with more conclusions (victories for either the player or the boss) than draws. It is necessary to explain that the concept of draw in Kromaia refers to a situation in which the game exceeds the maximum acceptable duration without concluding since with enough time one of the contenders will eventually win.

- **Duration (Viability):** Our approach calculates the criterion $Q_{Duration}$ as a measure of the average difference between the duration of each duel and the desired, optimal duration:

$$Q_{Duration} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels}\Big) \tag{3.6}$$

$Q_{Duration}$ is close to 1 when the duration of the duels is similar to an optimal value

T_d = *Duration of the d-th Duel*
$T_{Optimal}$ = *Optimal duration of a duel*

The duration of duels between players and boss units is expected to be around a certain optimal value. Significant deviations from that reference value are good design-flaw indicators: remarkably short games are probably too easy, thus minimizing the challenge offered by the game experience; duels that go on a lot longer than expected tend to make players lose interest. Duration is a simple and consistent criterion that is effective at determining essential game design proficiency and is one of the criteria used by Cardamone [13].

- **Uncertainty (Quality):** 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.

$$Q_{Uncertainty} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min\,(P_d, B_d)}{T_d}}{Duels}\Big) \tag{3.7}$$

$Q_{Uncertainty}$ is high when the contenders keep safe in terms of health as long as possible during the duels

P_d = *Time until Player health is critical in the d-th duel*
B_d = *Time until Boss health is critical in the d-th duel*

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 health levels of the player or the boss unit reach a dangerous/critical status (P_d and B_d, respectively).

- **Killer Moves:** Q_{KMoves} measures the proportion of killer moves by any contender, taking into account the moves done by both the player and the boss unit that are considered remarkable highlights but that are less important than killer moves.

$$Q_{KMoves} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels}\Big) \tag{3.8}$$

Q_{KMoves} reaches a high value when the percentage of highlights of duels with contenders that are not close is low

K_d = Number of killer moves detected in the d-th duel
H_d = Number of highlights detected in the d-th duel

This criterion is related to the fact that some events allow one of the contenders in a duel to make a remarkable impact in terms of power balance. They are the result of actions carried out on purpose that cause such an important effect, but that is not decisive enough to end the duel. In this video game, the developers considered that a highlight move occurs when either the boss unit or the player experiences a health decrease, but that killer moves are those that make the health difference between the contenders reach 30%. The health level of a boss depends on the number of weak points left in a specific unit, whereas a player can absorb five impacts.

- **Permanence:** The criterion $Q_{Permanence}$ is measured as follows:

$$Q_{Permanence} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d + K_d}}{Duels}\Big) \tag{3.9}$$

$Q_{Permanence}$ is higher when the advantages provided by killer moves or highlights are not canceled often

R_d = Number of recovery moves detected in the d-th duel

Duels with a high permanence value are games in which the advantage 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. Another move considered by the developers is the recovery move (R). This move quickly cancels the advantage given by other previous killer or highlight moves.

- **Lead Change:** This criterion is measured taking into account those highlights or killer moves that cause the lead to change during the course of a duel:

$$Q_{LChange} = clamp_{[0,1]}\Big(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d + K_d}}{Duels}\Big) \tag{3.10}$$

$Q_{LChange}$ gets closer to 1 as the number of relevant events which cause lead changes is higher in the duels

L_d = Number of lead changes detected in the d-th duel

In the video game case study, the lead is defined at any given moment by determining the contender with the highest health level, and a low number of lead changes indicates low dramatic value.

Our approach evaluated these six criteria for each boss unit 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.

3.5.3 Experimental Setup

The main goal of the evaluation was to measure the performance achieved by our EBI approach compared to the completely configured bossed included in the commercial release of Kromaia in terms of the six quality metrics proposed. To do this, we follow four steps:

The **first step** is the extraction of bosses from Kromaia, the video game case study. Using their version control system, the developers provided the following: the five partially configured bosses that will be used as input for the EBI approach; the five completely configured versions of the bosses that will be used in the comparison in terms of quality; the time spent in the configuration of the five original bosses.

The **second step** is the execution of the EBI approach. It receives a partially configured boss and produces one completely configured boss. This is repeated for the five partially configured bosses previously extracted from Kromaia. Given the stochastic nature of the approach, we perform 30 independent runs of the approach for each boss (as suggested in [3]), to homogenize the results and ensure that the evolutionary algorithm produces results that are consistent and repeatable.

The **third step** is the comparison of the completely configured bosses produced by the EBI approach and the bosses obtained from Kromaia. Therefore, the six quality measures are calculated for both sets of bosses. To do this, 30 duel simulations between the AI player and each of the bosses are performed and the quality measures are calculated. The results are gathered, averaged, and compared using box plots.

The **fourth step** is the statistical analysis of the results (following the guidelines in [2]), which provides quantitative evidence of the impact of the results and shows whether this impact is significant. Since our data does not follow a normal distribution, our analysis requires the use of non-parametric techniques. We carry out a Quade Test [18] followed by a Holm's post hoc to determine if the differences between pairs of bosses (one from EBI and one from Kromaia) are significant enough to be considered different. Then, we apply Vargha and Delaney's effect size [58] to determine to what degree the generated boss is better than the original boss for each of the six quality measures applied.

3.5.4 Implementation Details

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

The parameter settings used in our approach are detailed in Table I. The size of the population is limited to 100 individuals. Each generation, an elitism of 55 individuals is applied. The best 10 parents (μ) are selected by truncation to generate offspring of 45 new individuals through crossover and mutation of pairwise combinations of the parents. The probability of mutation (p_m) depends on the size of the individual (`1/(Number of Hulls)`

All of the parameters displayed in Table 3.1 were obtained from different tests. These tests showed that the settings currently applied reached better solutions in less time. In this work, we do not focus on tuning the parameters to achieve higher performance numbers for specific problems. In the context of testing, the default values used to measure the performance of search-based techniques are good enough, as suggested by Arcuri and Fraser [3]. Nevertheless, we intend to evaluate the parameters of our EBI approach in future works.

Table 3.1: EBI Approach Parameters

Parameter description	**Value**
Size: Population size	100
μ: Number of parents	10
λ: Number of offspring from μ parents	45
p_m: Mutation probability	1 / (Number of Hulls)

In general, there are two (atomic) performance measures used in search algorithms: a measure for speed or search effort, and a quality measure. In this paper, after running some prior convergence tests, we established a certain amount of wall clock time for each of the runs of our EBI approach (1200 seconds). Then, we focused on the solution quality.

For replication purposes, the CSV files used as input to report the results and the statistical analysis are available at: `https://svit.usj.es/bosses-kromaia-vs-ebi/`. There is a CSV file per boss in Kromaia where each file includes the results of each quality measure. There is also a CSV per boss configured in EBI where each file includes the mean results of the 30 runs from the best *Size* (100) configurations for each quality measure.

3.5.5 Results

This subsection presents five completely configured bosses obtained from our approach and compares them to the five bosses included in the commercial release of Kromaia using six quality measures studied in the video game research literature [12] and statistical methods.

Fig. 3.6 shows the results in different box plot groups representing the study of each quality measure. For every quality measure, the values belong to the interval [0,1] and the box plots represent the average results obtained (after 30 runs, due to non-deterministic factors) by each of the completely configured bosses generated by our approach (`Boss1..5`). In addition, every box plot includes a red diamond showing the average quality results achieved by the corresponding boss included in the commercial release of Kromaia for each quality measure.

Table 3.2 shows the *p-Values* of Holm's post hoc analysis for each boss and measure. A *p-Value* under 0.05 is statistically significant as accepted by the research community [2]. Each row of the table shows the results of each pair-wise comparison between a boss from Kromaia and a boss from our EBI approach, whereas the columns of the table show the Holm's post hoc *p-Values*. For example, the row "K1 vs E1" shows the *p-Values* of Holm's post hoc analysis for each measure that correspond to the pair-wise comparison between Boss 1 of Kromaia (K1) and Boss 1 of our EBI

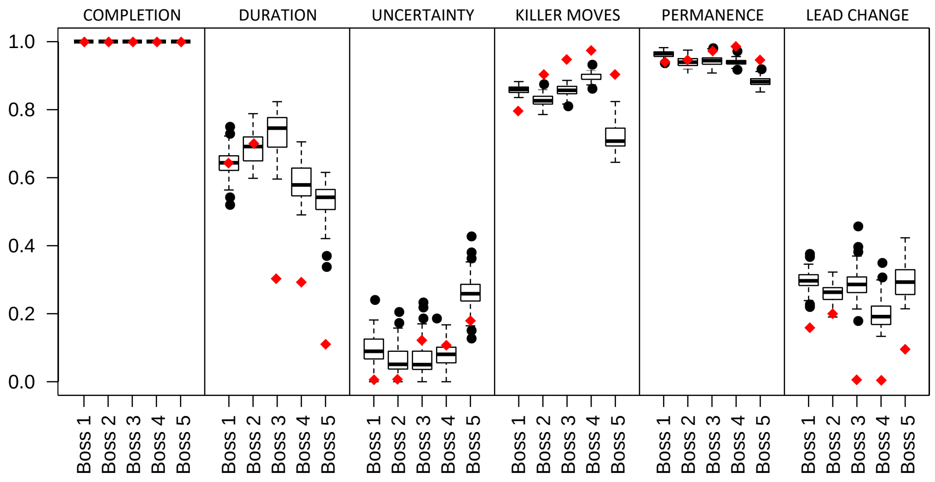


Figure 3.6: Results for the bosses generated by our approach for each quality measure. The columns show red diamonds representing the values obtained by the bosses originally included in the video game case study.

approach (E1). Table 3.3 shows $\hat{A}_{12}$ values for each boss and measure. For example, the row "K1 vs E1" shows $\hat{A}_{12}$ values for each measure that correspond to the pair-wise comparison between K1 and E1.

Table 3.2: Holm's post hoc *p-Values* for each boss and measure

	Holm's post hoc *p-Values*					
	Completion	Duration	Uncertainty	Killer Moves	Permanence	Lead Change
K1 vs E1	-	0.53	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K2 vs E2	-	0.21	0.36	$\ll 2 \times 10^{-16}$	2.8×10^{-11}	$\ll 2 \times 10^{-16}$
K3 vs E3	-	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K4 vs E4	-	$\ll 2 \times 10^{-16}$	1.3×10^{-9}	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K5 vs E5	-	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$

Table 3.4 shows the results per quality measure by taking into account the mean for that quality measure in all bosses. Each row of the table corresponds to a quality measure. Column 2 shows the standard deviations of the mean results of Kromaia bosses for each measure, whereas Column 3 shows the standard deviations of the mean result of the bosses

Table 3.3: $\hat{A}_{12}$ effect size for each boss and measure

	Completion	Duration	Uncertainty	Killer Moves	Permanence	Lead Change
K1 vs E1	0.5	0.48	0.01	0	0.04	0
K2 vs E2	0.5	0.52	0.59	1	0.76	0.02
K3 vs E3	0.5	0	0.91	1	1	0
K4 vs E4	0.5	0	0.77	1	1	0
K5 vs E5	0.5	0	0.06	1	1	0

of our EBI approach. Column 4 (Holm's post hoc *p-Values*) and Column 5 ($\hat{A}_{12}$) show the results of the pair-wise comparison between the mean of the Kromaia bosses and the mean of the EBI bosses for each quality measure. Each of the bosses produced by our approach is obtained by using the partially generated version of one of the original bosses included in Kromaia as a starting point. This partial generation initially constrains the characteristics of the bosses produced by our approach to a certain extent. For example, it determines the number of hulls used, which is not modified via encoding during the Equipment Configuration stage. The restrictions resulting from the partial generation do not determine the quality of the bosses, but characteristics like anatomy or behavior are affected. Therefore, the pair-wise comparison is used in order to study the differences between bosses in terms of the Equipment Configuration stage. The mean quality values for both the bosses included in Kromaia and those produced by our approach are also summarized in the radar chart included in Fig. 3.7.

3.5.6 Research Question 1

Does our EBI approach provide completely configured bosses that are comparable (or even better) in quality to those designed by the developers of the video game case study?

To answer the first research question, we compared the quality values achieved by the completely configured bosses obtained from our approach and the original bosses in the commercial release of the video game case study:

Table 3.4: For each measure, mean results and standard deviations, Holm's post hoc *p-Values* and $\hat{A}_{12}$ effect size

	Mean $\pm$ (σ)		Kromaia vs EBI Bosses	
	Kromaia Bosses	EBI Bosses	Holm's post hoc *p-Values*	$\hat{A}_{12}$
Completion	1 ± 0	1 ± 0	-	0.5
Duration	0.41 ± 0.25	0.64 ± 0.02	$\ll 2 \times 10^{-16}$	0
Uncertainty	0.10 ± 0.07	0.11 ± 0.02	1.2×10^{-10}	0.26
Killer Moves	0.91 ± 0.07	0.83 ± 0.01	$\ll 2 \times 10^{-16}$	1
Permanence	0.97 ± 0.02	0.93 ± 0.01	$\ll 2 \times 10^{-16}$	1
Lead Change	0.11 ± 0.07	0.27 ± 0.02	$\ll 2 \times 10^{-16}$	0

For **Completion** (the first column in Fig. 3.6), both the bosses from our approach and the original bosses obtained good values close to the maximum. These high values for $Q_{Completion}$ come from the characteristics of the battles between players and bosses in Kromaia: the effectiveness of the weapons used and the difficulty for an average player to avoid combat maximize the duel conclusions within acceptable duration limits.

For **Duration**, the more similar the duration of duels between players and bosses to the optimal value estimated, the higher the quality. The second column in Fig. 3.6 shows that the bosses generated by our approach outperformed the bosses in the case study. Monitoring the evolution of the Equipment Configuration population showed that it took few generations for the EA in our EBI approach to obtain $Q_{Duration}$ values that surpassed those achieved by the original bosses. The Version Control System used by the development company shows that the different Equipment Configurations tested varied little in terms of weapon/weak point sizes. This shows that, for $Q_{Duration}$, it is necessary to explore a larger Equipment Configuration search space than the one provided by the developers in order to obtain weapon and weak point distributions that achieve high values.

For **Uncertainty** (the third column in Fig. 3.6), four of the bosses generated by our approach achieved mean values similar to those obtained by the original bosses. High $Q_{Uncertainty}$ values would denote duels for

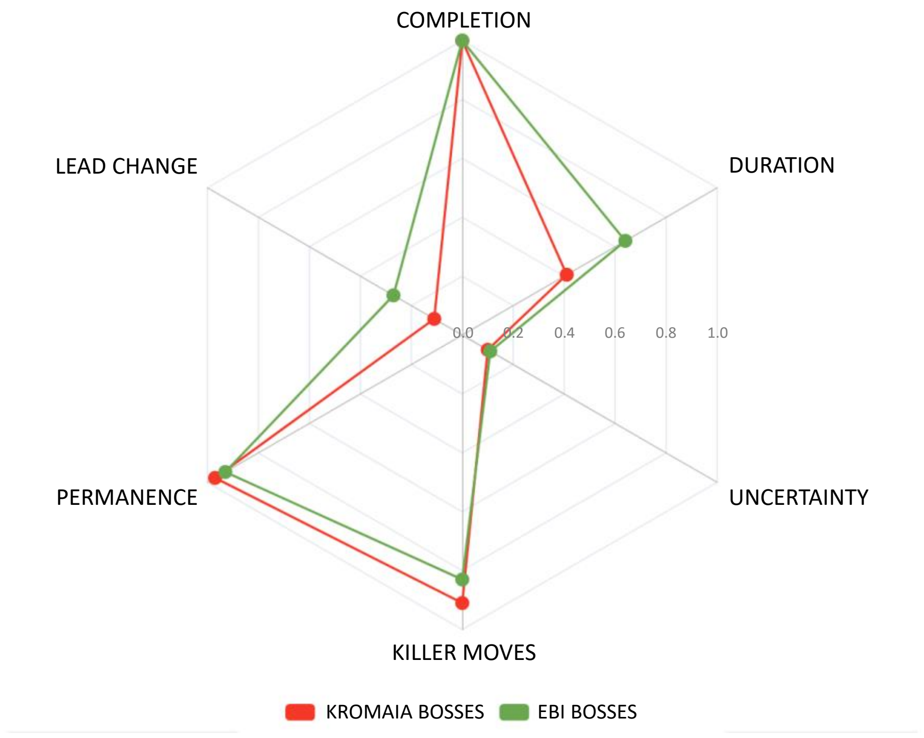


Figure 3.7: Quality measure mean values obtained by the bosses originally included in Kromaia, the video game case study, and those produced by our EBI approach.

which the time between when one of the contenders is about to be defeated and the conclusion is very low. In comparison to other video games, the design of Kromaia does not favor high uncertainty values. As duels advance, the bosses become more damaged and the weak point set is reduced. The distance in time between hits also progressively increases. For players, each time a health level decrease occurs, there is a safety time period during which the player is immune. The column displaying data related to Uncertainty in Fig. 3.6 shows that the fifth boss generated by our approach (`Boss5`) achieved a higher value than its corresponding boss. The most remarkable restriction in that fifth boss for both the commercial release of the video game case study and our approach is the fact that it is the boss that has the simplest geometry in terms of hull size (11) in comparison to the rest of the bosses, which have an average of 47 hulls. Therefore, considering the search spaces for the corresponding encoding sizes ($2^{11*5} = 2^{55}$ and $2^{47*5} = 2^{235}$, respectively), our algorithm managed to obtain a significantly higher value for `Boss5` with the time budget assigned.

For **Killer Moves** (Q_{KMoves}), the average values obtained by four of the bosses generated with our approach (`Boss1..4`) are comparable to those achieved by the five original bosses, as shown in Fig. 3.6. However, the fifth boss obtained by our approach (`Boss5`) achieved a lower mean value than the original boss. Due to the anatomical restrictions on hull size for both our approach and the original bosses, the version of the fifth boss generated by our approach obtained a higher percentage of uncertain duels, as described above. As shown in Fig. 3.6, there is an inverse relationship between Uncertainty and Killer moves: bosses with higher $Q_{Uncertainty}$ values obtain lower values for Q_{KMoves}. This is because a high number of killer moves (highlighted events that mean a significant health gap between contenders) is incompatible with high uncertainty.

For **Permanence** ($Q_{Permanence}$), the fifth column in Fig. 3.6 shows that the average values obtained from the bosses provided by our approach and those in the video game case study are comparable (even more so when `Boss5` is not considered). The reason behind that specific case is that Permanence measures the persistence in time of an advantage achieved in terms of health level gaps, so there is an inverse relationship between Q_{KMoves} and $Q_{Permanence}$.

For **Lead Change** ($Q_{LChange}$), the sixth column in Fig. 3.6 shows that the bosses generated by our EBI approach obtained better values than the original bosses included in Kromaia. Due to the specific design of the video game case study, it is difficult to achieve a high proportion of lead-change events in relation to highlights (health decrease events) or killer moves (highlights resulting in a considerable health gap between contenders). In Kromaia, a player can absorb up to five impacts before being defeated. However, the bosses generated by our approach and those included in the commercial release of Kromaia include a considerably higher number of weak points in order to reach acceptable values in the rest of the quality measures. Therefore, an impact received by a player is likely to take several weak points in the boss (each of which involves a highlight event) in order to revert the lead status. Since our approach obtains bosses with better $Q_{Duration}$ values, those bosses favor duels that are closer to the optimal duration and that have more lead changes. This minimizes the possibility of duels that are too long or too short, which occur due to passiveness or a significant skill difference between the contenders, respectively.

The $p\text{-}Values$ of Holm's post hoc analysis of Table 3.2 show that the differences in the bosses from Kromaia and the bosses from our EBI approach are statistically significant (under 0.05), except for the following: for Completion, the results are equivalent in all pair-wise comparisons of bosses; for Duration, the results are not significant when Bosses 1 and 2 are compared; and for Uncertainty, the results are not is not significant when Bosses 2 are compared. With regard to the $\hat{A}_{12}$ values of Table 3.3, Boss 1 from our EBI approach outperforms Boss 1 from Kromaia in all quality measures (except for Completion, which is equivalent), as row "K1 vs E1" of the table shows. The highest differences correspond to Uncertainty, Killer Moves, and Lead Change where Boss 1 of our EBI approach outperforms Boss 1 of Kromaia in almost all of the runs, respectively. This is especially relevant because Kromaia developers acknowledged that Boss 1 is the boss that they have devoted the most development effort to since Boss 1 is the most played and defeated boss in the video game (up to 6 times more than Boss 5 according to the global gameplay statistics of Steam).

When the data of bosses is aggregated by quality measure, the $p\text{-}Values$ of Holm's post hoc analysis of Table 3.4 show that the differences are statistically significant (under 0.05), except for Completion where the results are equivalent. The $\hat{A}_{12}$ values of Table 3.4 show that EBI bosses outperform Kromaia bosses in the majority of the runs for Duration, Uncertainty, and Lead Change. In Killer Moves and Permanence, Kromaia bosses outperform EBI bosses in all of the runs. Although it may appear that EBI bosses do not achieve good results for Killer Moves and Permanence, these results are comparable to the results achieved by the original Kromaia bosses. The difference between the mean of Killer Moves in Kromaia and Killer Moves in EBI is only 0.08; the difference between the mean of Permanence in Kromaia and Permanence in EBI is only 0.04. We showed these results to two Kromaia developers and they confirmed that the results obtained by our EBI approach are comparable (or even better) in quality to the results in Kromaia, so they will use our EBI approach to produce new, completely configured bosses in Kromaia through downloadable content.

3.5.7 Research Question 2

Does our EBI approach reduce the time necessary to configure good quality bosses in the video game case study?

This research question takes into account the time spent by the developers and our approach to perform the last configuration stage (Equipment Configuration) to obtain completely configured bosses that achieve good quality values.

To evaluate this last configuration stage, it was necessary to study the Version Control System of Kromaia used by the developers. The log history registry shows that the sum of the Equipment Configuration stages that led to the original bosses that were commercially released was five months. For our approach, the stop condition for the evolutionary algorithm in EBI was restricted only by time (20 minutes). Therefore, since it was necessary to generate five completely configured bosses, this task was completed in 1 hour and 40 minutes.

These results show that our approach generated good-quality, completely configured bosses about 99% faster than the developers. In addition, both the evolutionary algorithm and the duels that guide it and confront simulated contenders allow our approach to run unattended. Therefore, with better equipment, it would be possible to launch five separate instances of EBI (one per boss and CPU) in order to divide the time required by 5.

3.5.8 Fitness Progress in the Evolutionary Algorithm

With regard to the progress shown by the evolutionary algorithm used by EBI, Table 3.5 shows the average fitness values obtained by the best candidate of each of the five bosses generated by EBI at the end of the evolution. The fitness values of the five bosses from Kromaia (depicted as red diamonds in Fig. 3.6) are also included for comparison. The five EBI bosses reached mean fitness values of around 0.9. The fitness values shown by the original bosses included in Kromaia are low in comparison. Only the original K2 obtained values of 0.6 for both $F_{Victory}$ and F_{Health}. With regard to the convergence of the search performed by EBI, the fourth column in Table 3.5 shows the percentage of the time budget needed to find the best boss candidate. Interestingly, the improvement of the randomly-equipped original population used by the evolutionary algorithm is an average of 20% for three of the bosses and 10% for two of them. The last column of Table 3.5 shows the number of bosses explored during the time budget allocated (20 minutes). Additionally, doubling the time budget from 20 to 40 minutes did not affect the maximum fitness value reached).

Table 3.5: Average fitness values of the best candidate at the end of each evolution, percentage of the time budget required and number of bosses explored. Kromaia bosses are also included for comparison.

	Fitness			Time budget	
	$F_{Victory}$	F_{Health}	$F_{Overall}$	% needed	Bosses explored
E1	0.91	0.86	0.88	10%	305,650
K1	0.08	0.1	0.09		
E2	0.99	0.86	0.92	35%	566,640
K2	0.6	0.6	0.6		
E3	0.99	0.83	0.91	11%	523,855
K3	0.09	0.11	0.10		
E4	0.98	0.83	0.91	3%	311,140
K4	0.08	0.08	0.08		
E5	0.99	0.86	0.93	0.1%	1,420,165
K5	0.08	0.09	0.08		

3.6 Threats to Validity

In this section, we present how we addressed or mitigated the possible threats to validity regarding our approach. To identify the threats to be considered in this work, we use the guidelines described by De Oliveira et al. [14] and classify the threats into different groups.

Conclusion validity threats are concerned with the statistical relationships between data treatment and outcome. We have identified the following two threats in this category. The first one is not accounting for random variation. We addressed this threat by performing 30 runs for each of the bosses to be configured with our approach. The second one is the lack of a meaningful comparison baseline. To address this threat, we compared the results obtained from our approach with those generated by the developers for the commercial release of the video game case study.

Internal validity threats involve non-causal relationships between treatment and outcome. We have identified the following two threats in this category. The first one is the lack of clarity of data collection tools and procedures. Since it is difficult to calculate fitness values based on tests with players (due to their considerable time length), our approach simulates duels between the bosses and an AI player. We used the data provided by the SDMLs of the contenders to perform the simulation, and we used two main indicators provided by the developers to value configurations: victory percentage and health level. The second threat is the lack of real problem instances. To address this, the evaluation performed in our work was applied to an industrial video game case study, and the problem artifacts (partially and completely configured bosses) were directly obtained from the video game industry.

Construct validity threats are concerned with the relations between observations and theory. We have identified the following threats in this category. The first one is not assessing the validity of cost measures. In order to perform a fair comparison between the completely configured bosses included in Kromaia and the bosses generated by our approach, we studied the time spent by the developers and our algorithm to obtain the results (see Section 3.5.7). The second threat is not assessing the validity of effectiveness measures. We addressed this by using quality measures presented in the video game research literature [12] and performing a statistical analysis of the results (see Sections 3.5.2 and 3.5.5).

External validity threats deal with the generalization of the results obtained in a larger population, which is outside the experiment. We have identified one threat in this category, the lack of a clear definition of target instances. To address this threat, we have provided as much detail as possible regarding the DSL used by the video game case study (SDML, see Section 3.3.2). Nevertheless, EBI should be applied to other video games before assuring its generalization.

3.7 Conclusion

The creation of video game content is relevant to user retention and engagement. Our work shows that it is possible to accelerate video game content creation by means of Procedural Content Improvement. In this work, we focus on game bosses and the improvement of their quality through procedural content generation using our EBI approach. It works with an evolutionary algorithm that is guided by the simulation of duels between a player and the game bosses produced. The evolutionary algorithm and the genetic operations which we propose in our approach give priority to the content fragment which must be added to the initial fragment in order to consider such content complete. However, the improvement is not limited to that last fragment of the content, and these algorithm and operations drive a genetic improvement of the complete content, including the fragment that is provided originally as partially complete content.

We evaluate our approach in the context of Kromaia, a commercial PC and PlayStation 4 video game. In the application of the EBI to this game, our approach receives partially created bosses and produces complete bosses by automatically performing the last step in the creation process followed by human developers: the Equipment Configuration stage. This stage adds weapon and weak point content and defines the characteristics of a boss in terms of attack/defense items and weak point distribution. In the evaluation of our approach, we use six quality indicators from the video game research literature in order to give the game bosses a quality level value: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. We use these measures to evaluate the quality of both the bosses produced by our approach and those originally created by the developers of Kromaia.

The results show that our EBI approach provides completely configured bosses that are comparable (or even better) in quality to those designed by the developers of Kromaia. Our EBI approach improves video game content in less time than the developers of a commercial video game. The improvement of this content is an essential issue that concerns developers due to the nature of life cycles in commercial video games. This includes renovating products through downloadable content. For future work, we are planning to explore the potential benefits of EBI for level improvement.

Declarations

Funding. This work was supported in part by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the Project VARIATIVA under Grant PID2021-128695OB-I00, and in part by the Gobierno de Aragon (Spain) (Research Group S05_20D).

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Bibliography

[1] Ingo Althöfer. “Computer-Aided Game Inventing”. In: *Technical Report, Friedrich Schiller Universität Jena* (2003) (cit. on p. 147).

[2] Andrea Arcuri and Lionel Briand. “A Hitchhiker’s Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering”. In: *Softw. Test. Verif. Reliab.* 24.3 (May 2014), pp. 219–250. ISSN: 0960-0833. DOI: 10.1002/stvr.1486 (cit. on pp. 151, 153).

[3] Andrea Arcuri and Gordon Fraser. “Parameter tuning or default values? An empirical investigation in search-based software engineering”. In: *Empirical Software Engineering* 18.3 (2013), pp. 594–623. ISSN: 1573-7616. DOI: `10.1007/s10664-013-9249-9` (cit. on pp. 144, 151, 152).

[4] D. R. Ashley et al. “Learning to Select Mates in Evolving Non-playable Characters”. In: *2019 IEEE Conference on Games (CoG)*. 2019, pp. 1–8 (cit. on p. 123).

[5] Marlene Beyer et al. “An integrated process for game balancing”. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2016, pp. 1–8 (cit. on p. 127).

[6] Aditya Bhatt et al. “Exploring the hearthstone deck space”. In: *Proceedings of the 13th International Conference on the Foundations of Digital Games*. 2018, pp. 1–10 (cit. on p. 128).

[7] Daniel Blasco, Carlos Cetina, and Oscar Pastor. “A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study”. In: *Information and Software Technology* 119 (2020). ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2019.106235` (cit. on p. 125).

[8] Daniel Blasco et al. “An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering”. In: *Journal of Systems and Software* 171 (2021), p. 110804. ISSN: 0164-1212. DOI: `https://doi.org/10.1016/j.jss.2020.110804` (cit. on p. 125).

[9] Ilhem Boussaïd, Patrick Siarry, and Mohamed Ahmed-Nacer. “A survey on search-based model-driven engineering”. In: *Automated Software Engineering* 24.2 (2017), pp. 233–294 (cit. on p. 136).

[10] Joseph Alexander Brown et al. “Evolutionary Graph Compression and Diffusion Methods for City Discovery in Role Playing Games”. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8. DOI: `10.1109/CEC48606.2020.9185601` (cit. on p. 123).

[11] Cameron Browne. *Connection Games: Variations on a Theme*. Natick, Massachussetts: AK Peters, 2005. ISBN: 1568812248 (cit. on p. 147).

[12] Cameron Browne and Frédéric Maire. “Evolutionary Game Design”. In: *IEEE Trans. Comput. Intellig. and AI in Games* 2.1 (2010), pp. 1–16. DOI: 10.1109/TCIAIG.2010.2041928 (cit. on pp. 121, 146, 147, 153, 163).

[13] Luigi Cardamone et al. “Evolving interesting maps for a first person shooter”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2011, pp. 63–72 (cit. on pp. 124, 148).

[14] Márcio De Oliveira Barros and Arilo Cláudio Dias-Neto. “0006/2011-Threats to Validity in Search-based Software Engineering Empirical Studies”. In: *RelaTe-DIA* 5.1 (2011) (cit. on p. 162).

[15] Omar Delarosa et al. “Mixed-initiative level design with rl brush”. In: *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer. 2021, pp. 412–426 (cit. on p. 126).

[16] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003 (cit. on p. 138).

[17] Epic Games. *Unreal Engine, Version 2018.3.9*. Cary, North Carolina, 1998 (cit. on p. 137).

[18] Salvador García et al. “Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power”. In: *Information Sciences* 180.10 (2010). Special Issue on Intelligent Distributed Information Systems, pp. 2044–2064. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2009.12.010 (cit. on p. 151).

[19] Matthew Guzdial, Nicholas Liao, and Mark Riedl. “Co-creative level design via machine learning”. In: *arXiv preprint arXiv:1809.09420* (2018) (cit. on p. 126).

[20] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *arXiv preprint arXiv:1704.03477* (2017) (cit. on p. 126).

[21] Mark Hendrikx et al. “Procedural Content Generation for Games: A Survey”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 9.1 (Feb. 2013).

ISSN: 1551-6857. DOI: 10.1145/2422956.2422957 (cit. on pp. 120, 123, 145).

[22] Hiroyuki Iida et al. "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. DOI: 10.1007/978-3-540-28643-1_41 (cit. on p. 147).

[23] Alexander Jaffe et al. "Evaluating competitive game balance with restricted play". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* Vol. 8. 1. 2012 (cit. on p. 127).

[24] Daniel Karavolos, Antonios Liapis, and Georgios N Yannakakis. "Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model". In: *Proceedings of the 13th international conference on the Foundations of digital games.* 2018, pp. 1–10 (cit. on p. 128).

[25] Stuart Kent. "Model Driven Engineering". In: *Integrated Formal Methods.* Ed. by Michael Butler, Luigia Petre, and Kaisa Sere. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 286–298. ISBN: 978-3-540-47884-3 (cit. on p. 125).

[26] Stuart Kent. "Model driven engineering". In: *International conference on integrated formal methods.* Springer. 2002, pp. 286–298 (cit. on p. 137).

[27] John R Koza. "Genetic programming as a means for programming computers by natural selection". In: *Statistics and computing* 4.2 (1994), pp. 87–112 (cit. on p. 120).

[28] Wolfgang Kramer. "What Makes a Game Good?" In: *The Games Journal* (2000) (cit. on p. 147).

[29] W. B. Langdon and M. Harman. "Optimizing Existing Software With Genetic Programming". In: *IEEE Transactions on Evolutionary Computation* 19.1 (2015), pp. 118–135. ISSN: 1089-778X. DOI: 10.1109/TEVC.2013.2281544 (cit. on p. 120).

[30] Pier Luca Lanzi, Daniele Loiacono, and Riccardo Stucchi. "Evolving maps for match balancing in first person shooters". In: *2014 IEEE Conference on Computational Intelligence and Games*. IEEE. 2014, pp. 1–8 (cit. on p. 124).

[31] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. "Sentient sketchbook: Computer-aided game level authoring". In: *In Proceedings of ACM Conference on Foundations of Digital Games, 2013. In Print* (cit. on p. 126).

[32] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. "Sentient world: Human-based procedural cartography". In: *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer. 2013, pp. 180–191 (cit. on p. 126).

[33] Siew Mooi Lim et al. "Crossover and mutation operators of genetic algorithms". In: *International journal of machine learning and computing* 7.1 (2017), pp. 9–12 (cit. on p. 136).

[34] Jialin Liu et al. "Deep learning for procedural content generation". In: *Neural Computing and Applications* 33.1 (2021), pp. 19–37 (cit. on pp. 126, 127).

[35] Ziwei Liu et al. "Deep learning face attributes in the wild". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3730–3738 (cit. on pp. 122, 126).

[36] Daniele Loiacono and Luca Arnaboldi. "Fight or flight: Evolving maps for cube 2 to foster a fleeing behavior". In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. 2017, pp. 199–206. DOI: 10.1109/CIG.2017.8080436 (cit. on p. 124).

[37] Daniele Loiacono and Luca Arnaboldi. "Multiobjective Evolutionary Map Design for Cube 2: Sauerbraten". In: *IEEE Transactions on Games* 11.1 (2019), pp. 36–47. DOI: 10.1109/TG.2018.2830746 (cit. on p. 124).

[38] Fernando de Mesentier Silva et al. "AI as Evaluator: Search Driven Playtesting of Modern Board Games." In: *AAAI Workshops*. 2017 (cit. on p. 127).

[39] Fernando de Mesentier Silva et al. "Evolving the hearthstone meta". In: *2019 IEEE Conference on Games (CoG)*. IEEE. 2019, pp. 1–8 (cit. on p. 128).

[40] Alex Pantaleev. “In search of patterns: Disrupting rpg classes through procedural content generation”. In: *Proceedings of the The third workshop on Procedural Content Generation in Games*. 2012, pp. 1–5 (cit. on p. 128).

[41] Kyungjin Park et al. “Generating educational game levels with multistep deep convolutional generative adversarial networks”. In: *2019 IEEE Conference on Games (CoG)*. IEEE. 2019, pp. 1–8 (cit. on p. 127).

[42] G Pavai and TV Geetha. “A survey on crossover operators”. In: *ACM Computing Surveys (CSUR)* 49.4 (2016), pp. 1–43 (cit. on p. 136).

[43] Francisca Pérez, Tewfik Ziadi, and Carlos Cetina. “Utilizing automatic query reformulations as genetic operations to improve feature location in software models”. In: *IEEE Transactions on Software Engineering* (2020) (cit. on p. 136).

[44] Justyna Petke et al. “Genetic Improvement of Software: A Comprehensive Survey”. In: *IEEE Trans. Evol. Comput.* 22.3 (2018), pp. 415–432. DOI: `10.1109/TEVC.2017.2693219` (cit. on p. 120).

[45] Johannes Pfau et al. “Dungeons & replicants: automated game balancing via deep player behavior modeling”. In: *2020 IEEE Conference on Games (CoG)*. IEEE. 2020, pp. 431–438 (cit. on p. 128).

[46] Emanuel Montero Reyno and José Á Carsí Cubel. “Automatic prototyping in model-driven game development”. In: *Computers in Entertainment (CIE)* 7.2 (2009), pp. 1–9 (cit. on p. 137).

[47] AndréSiqueira Ruela and Frederico Gadelha Guimarães. “Procedural generation of non-player characters in massively multiplayer online strategy games”. In: *Soft Computing* 21.23 (2017), pp. 7005–7020. DOI: `10.1007/s00500-016-2238-3` (cit. on p. 123).

[48] Anurag Sarkar and Seth Cooper. “Blending Levels from Different Games using LSTMs.” In: *AIIDE Workshops*. 2018 (cit. on p. 126).

[49] Anurag Sarkar, Zhihan Yang, and Seth Cooper. “Controllable level blending between games using variational autoencoders”. In: *arXiv preprint arXiv:2002.11869* (2020) (cit. on p. 126).

[50] Ygor Rebouças Serpa and Maria Andréia Formico Rodrigues. "Towards machine-learning assisted asset generation for games: A study on pixel art sprite sheets". In: *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE. 2019, pp. 182–191 (cit. on p. 126).

[51] Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016 (cit. on p. 145).

[52] Kristin Siu, Eric Butler, and Alexander Zook. "A programming model for boss encounters in 2d action games". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 12. 1. 2016 (cit. on p. 143).

[53] Sam Snodgrass and Anurag Sarkar. "Multi-domain level generation and blending with sketches via example-driven BSP and variational autoencoders". In: *International Conference on the Foundations of Digital Games*. 2020, pp. 1–11 (cit. on p. 126).

[54] Adam Summerville et al. "Procedural Content Generation via Machine Learning (PCGML)". In: *IEEE Trans. Games* 10.3 (2018), pp. 257–270. DOI: `10.1109/TG.2018.2846639` (cit. on p. 120).

[55] Stephen Tang and Martin Hanneghan. "A model-driven framework to support development of serious games for game-based learning". In: *2010 Developments in E-systems Engineering*. IEEE. 2010, pp. 95–100 (cit. on p. 137).

[56] J. Mark Thompson. "Defining the Abstract". In: *The Games Journal* (2000) (cit. on p. 147).

[57] Julian Togelius et al. "Search-Based Procedural Content Generation: A Taxonomy and Survey." In: *IEEE Trans. Comput. Intellig. and AI in Games* 3.3 (2011), pp. 172–186 (cit. on pp. 120, 122, 123, 129, 145).

[58] András Vargha and Harold D. Delaney. "A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong". In: *Journal of Educational and Behavioral Statistics* 25.2 (2000), pp. 101–132. DOI: `10.3102/10769986025002101`. eprint: `http://jeb.sagepub.com/content/25/2/101.full.pdf+html` (cit. on p. 152).

[59] Jan Salvador van der Ven et al. "Design Decisions : The Bridge between Rationale and Architecture". In: 2006 (cit. on p. 126).

[60] Vanessa Volz, Günter Rudolph, and Boris Naujoks. "Demonstrating the feasibility of automatic game balancing". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 2016, pp. 269–276 (cit. on p. 127).

[61] Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. "Mixed-initiative co-creativity". In: (2014) (cit. on p. 126).

[62] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*. Vol. 2. Springer, 2018 (cit. on pp. 129, 145).

[63] Byungho Yoo and Kyung-Joong Kim. "Changing video game graphic styles using neural algorithms". In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2016, pp. 1–2 (cit. on p. 126).

[64] Peter Thorup Ølsted, Benjamin Ma, and Sebastian Risi. "Interactive evolution of levels for a competitive multiplayer FPS". In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2015, pp. 1527–1534. DOI: `10.1109/CEC.2015.7257069` (cit. on pp. 124, 125).

Part III

Discussion

Discussion

The present part discusses this work, taking into consideration the motivation, research questions, and results of the thesis, given the outcomes of the research articles presented in chapters 1, 2, and 3 of Part II. Additionally, the discussion introduces the research lines of the new works started as a continuation of this research.

Research Questions

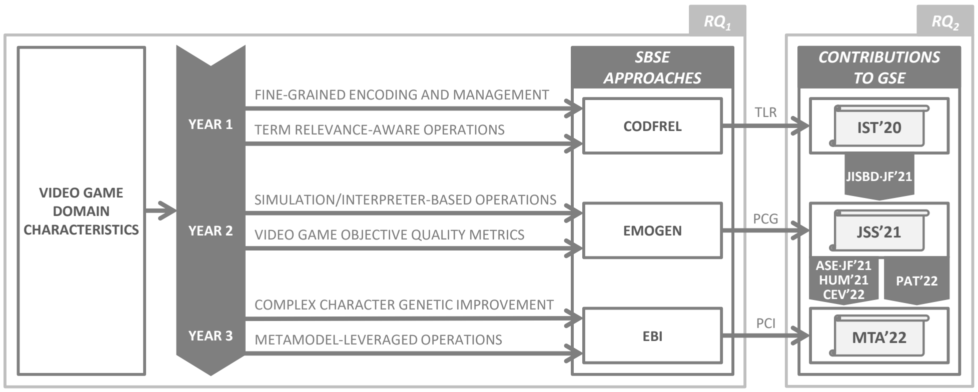

RQ1
RQ2
VIDEO GAME DOMAIN CHARACTERISTICS
YEAR 1
YEAR 2
YEAR 3
FINE-GRAINED ENCODING AND MANAGEMENT
TERM RELEVANCE-AWARE OPERATIONS
SIMULATION/INTERPRETER-BASED OPERATIONS
VIDEO GAME OBJECTIVE QUALITY METRICS
COMPLEX CHARACTER GENETIC IMPROVEMENT
METAMODEL-LEVERAGED OPERATIONS
SBSE APPROACHES
CODFREL
EMOGEN
EBI
TLR
PCG
PCI
CONTRIBUTIONS TO GSE
IST'20
JISBD-JF'21
JSS'21
ASE-JF'21 HUM'21 CEV'22
PAT'22
MTA'22

Figura 8: Thesis work progress summary.

As stated in the articles included in this thesis, there is a strong demand for approaches that increase the productivity with regard to maintenance and content production in the context of video game development. The growth of both the industry and the distribution platforms has led to a need for higher competitiveness, too. This thesis aims to contribute to industrial development through GSE by answering the following research questions:

RQ$_1$: *How can specific video game characteristics be used in order to develop SBSE approaches that address GSE problems?*

After studying the current status of the research communities of GSE, Game Software in general, and the possible benefits of leveraging SBSE in video games, TLR, and more specifically, Requirement Traceability, was selected as the starting research field, due to its importance. In addition, Kromaia, a commercial video game case study, was selected, due to its complexity, and the possibility of accessing the source code of both the game itself and the proprietary game engine used in its development, which includes a DSL that allows for the definition of characteristics such as characters, stages, or missions.

The first work addressed video game requirement traceability, taking into consideration the size of the source code in commercial video games and the level of dispersion of the requirements in such source codes, as well as the influence of the developers' domain knowledge with regard to term importance in natural language requirement descriptions. The evolutionary computation-based approach presented (CODFREL), produced results that were compared with those obtained by a baseline that did not take into account the video game specific aspects that were considered by CODFREL in the encoding and the genetic operations used, as shown in the top left part of Figure 8. The interest of the community made the publication of this work in a journal (**IST'20** [2]) possible.

The second work kept the use of evolutionary computation in order to search a large solution space, but the focus was another prominent area of interest for the Game Research community: PCG. More specifically, the goal was the generation of game boss software models that could be interpreted at run-time and translated into their source code equivalents. The EMoGen approach presented in the work included the use of game simulations with fitness purposes, as shown in the center part of Figure 8. These simulations consist in duels between two artificial agents (a human player and the boss evaluated) whose displacement, offensive, and health control actions are controlled by means of a state machine-based system. Additionally, the bosses produced were measured with widely accepted quality metrics from the Game Research literature. This work was accepted for its publication in a journal (**JSS'21** [3]).

The third work focused on the exploration, in more depth, of the issues studied in the second research work. The EBI approach that was developed in this work addressed PCI to generate complete game bosses, taking partially generated bosses as starting points. This third work was published in a journal (**MTA'22** [4]).

Thus, these works show that leveraging video game characteristics in order to develop SBSE approaches is beneficial for GSE topics.

RQ$_2$: *In case that video game focused SBSE approaches are successfully developed, what advances or improvements do they bring to GSE?*

With regard to the advances achieved, the outcomes of the research works of this thesis can be summarized as follows:

- In **IST'20** [2], the results produced by CODFREL outperformed the requirement realization solution candidates provided by a baseline that did not make use of the video game idiosyncrasy aspects considered by the approach proposed. The influence of tacit knowledge prevented the results achieved by CODFREL from being optimal.
- **JSS'21** [3] showed that the EMoGen approach managed to produce game boss software models that were comparable in quality with those created by the developers manually, and reduced the time required from months to hours. The results suggest that using characters that are not bosses as seeds could be useful in order to control the characteristics of the solution candidates and prevent them from being too similar to the existing game bosses included in the original game case study.
- In **MTA'22** [4], the EBI approach used PCI in order to produce complete content after receiving partially complete content as input. The results showed that EBI produced bosses with a quality comparable to that of the bosses entirely created by human developers. In addition, the time required was reduced from months to less than two hours.

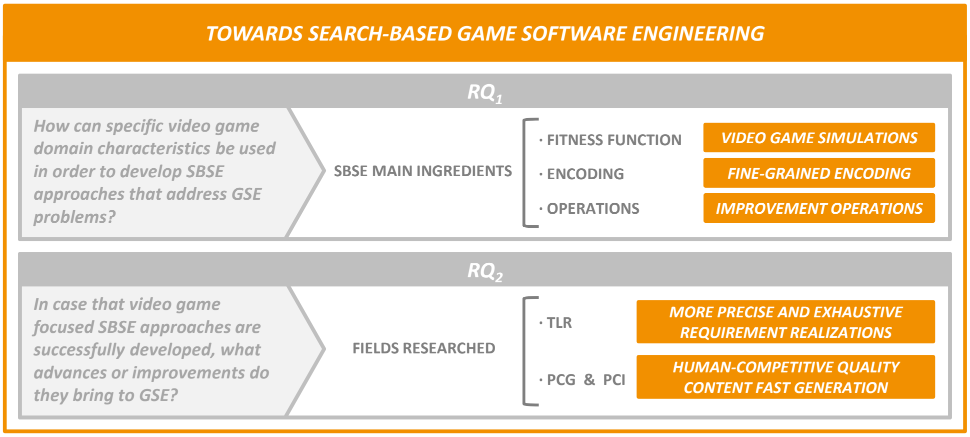

TOWARDS SEARCH-BASED GAME SOFTWARE ENGINEERING
RQ1
How can specific video game domain characteristics be used in order to develop SBSE approaches that address GSE problems?
SBSE MAIN INGREDIENTS
· FITNESS FUNCTION
· ENCODING
· OPERATIONS
VIDEO GAME SIMULATIONS
FINE-GRAINED ENCODING
IMPROVEMENT OPERATIONS
RQ2
In case that video game focused SBSE approaches are successfully developed, what advances or improvements do they bring to GSE?
FIELDS RESEARCHED
· TLR
· PCG & PCI
MORE PRECISE AND EXHAUSTIVE REQUIREMENT REALIZATIONS
HUMAN-COMPETITIVE QUALITY CONTENT FAST GENERATION

Figura 9: Research question outcome overview.

Figure 9 summarizes the outcomes of this thesis with regard to each of the research questions formulated. RQ$_1$ addresses SBSE design and development, which involves three core ingredients: a fitness function (the main ingredient), an encoding, and operations. The advances of this thesis in relation to those ingredients are, respectively: the use of video game simulations for measuring the value of solution candidates; fine-grained encoding as a response to highly disperse solution realizations in huge search spaces; and improvement operations which have an impact on non-encoded aspects of solution candidates. RQ_2 refers to the improvements achieved after using video game characteristics in SBSE, and the advances shown in the fields researched —TLR and PCG/PCI— are, respectively: solution candidates with higher levels of precision and thoroughness; and the generation of video game content that can be compared in quality with the creations of developers, but requiring a significantly lower time than that used by human domain experts.

Future Lines of Research

Video Game Simulation Evolution

In JSS'21 [3] and MTA'22 [4] video game simulations proved to be useful for generating video game content (specifically, video game characters) in order to obtain their different quality aspect values. A prominent attribute of simulations is the fact that they do not only represent actions taken by video game entities and their consequences, but also the way a certain game experience develops. From a video game design perspective, a successful game experience would be a compelling session that could be qualified as "entertaining" or "fun". Such properties are not easy to formalize through numerical data or textual descriptions and, due to that, simulations could be used in order to be more than fixed or static measurement methods, and evolve in order to search for interesting game experience traits or even locate bugs that are difficult to find due to their connection with the facets that make a experience good or bad for game designers and players. With purposes like those described, the simulations should be encoded to be the individuals that conform the population evolved, and therefore, representations such as parameter collections should be studied and discussed as potential valid encoding proposals, due to the complexity of a simulation in terms of character behaviour, among other main aspects.

Biological Systematics: Phylogenetics

In MTA'22 [4], it was possible to observe a fast convergence of potential solution candidates to high fitness values, which could prevent the evolutionary algorithm from visiting certain "areas" within the solution space. The works included in this thesis showed how the knowledge that is not explicitly available could lead to possible solutions that would otherwise remain unexplored. Such knowledge could be obtained from human developers, or formalized representations, like models, or by techniques that search for latent information, like LSI in IST'20 [2].

In line with the last of the options mentioned, an additional objective for my next works is the application of Biological Systematics, which studies the diversification of lifeforms [6, 5], to SBSE in GSE. More specifically, Phylogenetics [1] could be used in order to classify a population as if each individual represented a taxon (e.g., a species) and promote those potential solutions which belong to a desirable lineage. Phylogenetics could be applied in order to study the whole encoded genetic material of the individuals or just focus on aspects that are difficult to assess or classify, like the appearance or aesthetics of the game characters evolved. The use of techniques like Phylogenetic Tree Inference would extract the latent relationships of the population from the individuals and no additional knowledge bases or previous classification training would be required.

In the context of commercial video game development, a possible application scope could be the production of sequels within video game franchises: In video game series it is common to find characters, stages, or other game content items that are aligned with the content shown in previous installments, even if the new items introduce substantial novelties. The plan for the future includes studying the acceptance by domain experts of the game content generated using Phylogenetics.

Bibliography

[1] D.A. Baum and S.D. Smith. *Tree Thinking: An Introduction to Phylogenetic Biology.* Macmillan Learning, 2012. ISBN: 9781936221165 (cit. on p. 180).

[2] Daniel Blasco, Carlos Cetina, and Oscar Pastor. “A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study”. In: *Information and Software Technology* 119 (2020). ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2019.106235` (cit. on pp. 177, 178, 180).

[3] Daniel Blasco et al. “An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering”. In: *Journal of Systems and Software* 171 (2021), p. 110804. ISSN: 0164-1212. DOI: `https://doi.org/10.1016/j.jss.2020.110804` (cit. on pp. 177–179).

[4] Daniel Blasco et al. “Procedural content improvement of game bosses with an evolutionary algorithm”. In: *Multimedia Tools and Applications* 82 (2022), pp. 1–33. ISSN: 1573-7721. DOI: `https://doi.org/10.1007/s11042-022-13674-6` (cit. on pp. 177–180).

[5] Andrew VZ Brower and Randall T Schuh. *Biological systematics: principles and applications.* Cornell University Press, 2021 (cit. on p. 180).

[6] Charles Duncan Michener, National Research Council, et al. *Systematics in support of biological research.* National Academies, 1970 (cit. on p. 180).

Part IV

Conclusions

Conclusions

The last chapter describes the final conclusions of this thesis with regard to the objectives, results, and impact achieved by the research in the context studied through the development of the work.

Game Software Engineering (GSE) consists in the study and application of Software Engineering in the context of video game development. In the past literature, Game Software Research addressed topics like Maintenance and Content Generation, which are especially relevant due to the characteristics of the project life cycles associated to the video game industry. The work presented in this thesis aims to study such Game Software topics from a Software Engineering perspective in order to improve and accelerate game development. More specifically, this work studies Software Engineering optimization problems for which the generation of solutions requires the exploration of large search spaces. In addition, the literature shows how documentation or knowledge bases that could be used with training purposes are often not available. For those reasons, the research presented focuses on GSE through a field that has proven to be successful in problems like those studied: Search-based Software Engineering (SBSE).

In the past, the aforementioned relevant Game Software topics (Maintenance and Content Generation) were first studied outside GSE, but they are present in later GSE researches, too. However, GSE tended not to take into account video game domain knowledge in SBSE approaches to GSE for those issues: Video game source code architecture, requirement dispersion, or requirement description relevant terms, in the case of Requirement Traceability; and, in Content Generation, the characteristics of certain content, like complex game characters, the use of simulations in order to measure the value of such content, and the inclusion of video game domain quality indicators. This thesis explores those aspects that were neglected in the past GSE researches and, additionally, makes use of fields like Model-Driven Engineering in order to study the application of SBSE to GSE in areas that remained unexplored.

The surveys and the past works corresponding the fields covered by this thesis show that the researches usually involved video games that were small or academic projects, instead of medium or large scale commercial titles. This research focuses on industrial case study evaluations in order to better illustrate the potential benefits of the knowledge acquired for such a growing sector.

The research developed in this thesis has produced results that have been published as three articles in specialized and relevant journals with positive outcomes:

- The first work [1] focused on Requirement Traceability. The results showed that using a fine-grained approach and term relevance-aware guidance in genetic operations outperformed the techniques that were usually applied. Additionally, the results obtained evinced the importance of non-formalized tacit knowledge in the search for optimal solutions.
- The second work [2] studied Procedural Content Generation (PCG). The bosses obtained were comparable in quality to those manually created by the human developers, and the time required was significantly reduced. The results also showed how seeds which do not differ much from the original bosses of the case study were important for the production of higher quality results, in comparison with the use of random or non-boss character seeds.
- The third work [3] continued the exploration of PCG and, more specifically, Procedural Content Improvement (PCI) in cases in which a content partially generated is received and the final, complete content must be produced without the support of interpreters or reparation tools that could be used in order to guide the search in the solution space. The results produced showed that the complete bosses generated were produced with a significantly lower time budget and were comparable in quality to the bosses manually created, through all the different stages involved, by the human developers.

The results obtained in this research allow for presenting the following conclusions:

1. It is possible to improve Game Software Engineering by means of Search-based Software Engineering approaches that leverage video game characteristics. Such characteristics can be used in order to successfully guide searches in large solution spaces.
2. The successful application of Search-based Game Software Engineering approaches that take advantage of video game character-

istics leads to human-competitive results in terms of quality, and substantially reduces the time required to produce those results.

The research presented in this thesis reflects that there are still many questions to answer with regard to what knowledge could maximize the quality of the results obtained through Search-based GSE, and where that knowledge lies. The advances provided by this thesis contribute to the GSE research community with positive results and also introduce paths that could lead to answers to those open questions. In the end, the purpose of this work is not the promotion of automated searches over human developers, but helping developers with results that they could directly try, or use as starting points that might inspire them with new, interesting ideas to explore.

Bibliography

[1] Daniel Blasco, Carlos Cetina, and Oscar Pastor. "A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study". In: *Information and Software Technology* 119 (2020). ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2019.106235` (cit. on p. 187).

[2] Daniel Blasco et al. "An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering". In: *Journal of Systems and Software* 171 (2021), p. 110804. ISSN: 0164-1212. DOI: `https://doi.org/10.1016/j.jss.2020.110804` (cit. on p. 187).

[3] Daniel Blasco et al. "Procedural content improvement of game bosses with an evolutionary algorithm". In: *Multimedia Tools and Applications* 82 (2022), pp. 1–33. ISSN: 1573-7721. DOI: `https://doi.org/10.1007/s11042-022-13674-6` (cit. on p. 187).