Leveraging Phylogenetics in Software Product Families: The Case of Latent Content Generation in Video Games

Jorge Chueca
jchueca@usj.es
Universidad San Jorge
Zaragoza, Spain
Universitat Politècnica de València
Valencia, Spain

Daniel Blasco
dblasco@usj.es
Universidad San Jorge
Zaragoza, Spain

Carlos Cetina
ccetina@usj.es
Universidad San Jorge
Zaragoza, Spain

Jaime Font
jfont@usj.es
Universidad San Jorge
Zaragoza, Spain

ABSTRACT

A family of software products comprises similar products within a defined scope that share common characteristics, often due to reuse techniques applied during development. This paper introduces an approach that applies biological insights to map the landscape of a software product family, identifying potential gaps within its scope. Phylogenetics studies the gene similarity among groups of organisms to understand ancestry among species. Leveraging Phylogenetics in software, our approach offers a structured view of a product family, aiding in the discovery of unexplored areas fitting the scope of the family. Our approach creates a phylogenetic tree that enables to easily identify latent products (ancestors) that did not exist in the original family. Those ancestors can then be reconstructed from existing products (descendants). The product family evaluated is a set of industry-scale video game non-playable characters. We assess this approach through video game simulations and scope metrics to determine how closely the reconstructed products align with the family's scope. The results confirm that the content generated with phylogenetics aligns better with the family scope than the state-of-the-art procedural content generation techniques using evolutionary algorithms. Phylogenetics enhances content generation by providing a framework to understand and expand the product family with new content.

KEYWORDS

Phylogenetics, Software Product Families, Game Software Engineering, Procedural Content Generation

ACM Reference Format:
Jorge Chueca, Daniel Blasco, Carlos Cetina, and Jaime Font. 2024. Leveraging Phylogenetics in Software Product Families: The Case of Latent Content Generation in Video Games . In *Proceedings of 28th ACM International Systems and Software Product Lines Conference (SPLC'24).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software is mainly created through reuse. Since the term software engineering was coined at the NATO Conference held in Garmisch in 1968 [42], its evolution has been tied to the concept of reuse (also coined during that conference [38]). Either applying an opportunistic approach [36] such as clone-and-own, or applying a systematic approach as the software product lines propose [13].

These practices result in most products being built by reusing existing ones. Several studies [6, 25, 29, 30] have reported reuse percentages across products ranging from 10% to 85% during the late 80s. More recently, the plastic surgeon theory [4] found that 43% of commits to a large repository of Java projects could be reconstituted from existing code. Similarly, Gabel and Sue [17] concluded that to be able to write a new piece of software in a large repository (sourceforge), an engineer needs to write more than six lines of code; otherwise, the code already exists somewhere.

However, despite extensive reuse in software creation, the increasing demand for software pressures developers who struggle to meet expectations. This scenario is even worse in the case of Game Software Engineering (GSE) [1, 39], where creating new content for video games is the bottleneck in an industry that has become the largest entertainment sector, surpassing music and cinema [44] and accounting for one out of two software developers [54].

To address this need for content, some works from the GSE community have applied Procedural Content Generation (PCG) techniques [22, 56] to accelerate the development of new features for their families of products. However, those techniques are primarily based on evolutionary computation [50], and the generated elements often fall outside the intended scope due to their reliance on randomness. Alternatively, other works use Machine Learning for PCG (PCGML), but the lack of training content remains a challenge for PCGML [49]. Additionally, interpreting the results is challenging for game developers [35], who must create content that meets design expectations.

By contrast, we argue that the inherent reuse across a family of video game elements enables an analysis to generate new content by reusing parts of the family. In this work, we propose using phylogenetics, a nature-inspired approach from biology, to analyze ancestry among family products and exploit them to generate new products within the family scope.

Phylogenetic analysis is used in biology to discover ancestry relationships among individuals, classify them, and formalize an evolution tree that arranges them in a succession of evolutions. This technique yields the discovery of hidden links among the individuals that have yet to be discovered. We argue that this technique can be adapted to a family of software products, yielding the discovery of missing ancestors that can fit into the family of products.

To validate these ideas, we perform a phylogenetic analysis of a family of software products (video game elements from a commercial game), arrange the results into a phylogenetic tree, identify ancestors, reconstruct them using their descendants, and evaluate them as new family products. We then compare these reconstructed ancestors with procedurally generated elements using a state-of-the-art approach, assessing both sets with metrics used in video games to determine the alignment with the game's scope.

The results indicate that the elements generated with our approach align more closely with the original family of products' scope than those generated with the state-of-the-art approach, representing a smaller scope difference 75.88% of the time. The statistical analysis shows that the differences are significant, and there is a large effect size between our approach and the baseline. Applying the proposed technique can create new content and provide a new perspective on the video game's product line.

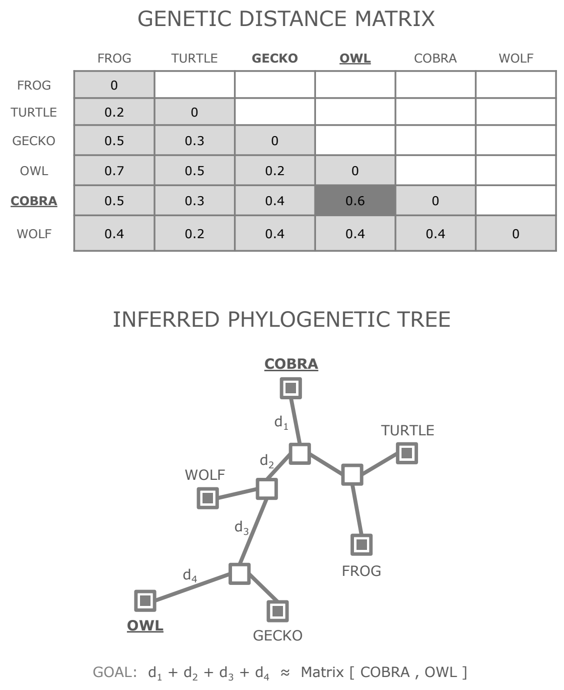
This study paves a new path in content generation. Previous approaches generate content that is considered secondary content (e.g., vegetation) because they are not aligned with the scope of video games. This is a new point of view for SPLs where genetic divergence points and ancestry branches are exploited. This new perspective opens new possibilities for understanding variability. For instance, Phylogenetics has the potential to initialize an SPL via re-engineering, among other uses.

The paper is structured as follows. Section 2 presents the background for this work. Section 3 presents the steps in our Latent Content Generation approach. Section 4 presents the evaluation. Section 5 answers the research questions and presents a discussion of the results. Section 6 presents the study's threats to validity. Section 7 presents the related work. Section 8 concludes the paper.

2 BACKGROUND

This section introduces Phylogenetic inference and its characteristics borrowed from the Biology field and how it is used to create phylogenetic trees in nature.

In Biology, it is possible to measure the genetic difference between two given species or even between groups of individuals of the same species. That difference is denoted as genetic distance [43], and, in such a field of study, the genetic distance is calculated for two specific taxa. Taxon is a concept that refers to a group of organisms that have a set of genetic characteristics in common. For example, the *Animalia* kingdom is a high-level, general taxon that includes all the animals known, whereas *Canis Lupus* (Grey Wolf) is a low-level taxon that refers to a specific species only. The comparison of a given set of taxa is commonly done using a data


GENETIC DISTANCE MATRIX

	FROG	TURTLE	**GECKO**	**OWL**	COBRA	WOLF
FROG	0					
TURTLE	0.2	0				
GECKO	0.5	0.3	0			
OWL	0.7	0.5	0.2	0		
COBRA	0.5	0.3	0.4	0.6	0	
WOLF	0.4	0.2	0.4	0.4	0.4	0

INFERRED PHYLOGENETIC TREE
COBRA
TURTLE
WOLF
FROG
OWL
GECKO
d_1
d_2
d_3
d_4
GOAL: $d_1 + d_2 + d_3 + d_4 \approx$ Matrix [COBRA , OWL]

Figure 1: Example of an inferred phylogenetic tree from a distance matrix.

structure known as a Genetic Distance Matrix: a square matrix that represents the genetic distance between each pair of taxa studied [52]. The top part of Figure 1 shows how the diagonal values are always zero in a matrix like that. Typically, due to its symmetrical nature, only the distances above or below the main diagonal are represented to hide redundant information. In addition, the genetic distances are usually values between zero and one [23].

It is possible to apply inference techniques to produce a diagram called Phylogenetic Tree [5], which shows how the taxa studied (the leaves of the tree) are related in terms of genetic divergence points and ancestry branches using inner nodes, that hypothesize fossil ancestors, and distance values included in the edges. The lower part of Figure 1, which uses an animal species-based example, shows that the objective is to provide additional and useful branching/lineage information while being coherent with the genetic distances used. Therefore, for two given taxa, the sum of the distance values of the path that connects them in the tree should match or be close to the value present in the distance matrix.

3 LATENT CONTENT GENERATION

Our approach is based on the new perspective that phylogenetics can provide to a product family by representing the genetic relations among products in a single tree. From this tree, the game developer can identify potential gaps (ancestors) that serve as a seed for creating new content. We call this new content as Latent Content.

Figure 2 represents an overview of our approach. Dotted boxes represent artifacts generated and solid line boxes represent execution actions or calculations. The LCG approach is divided into two

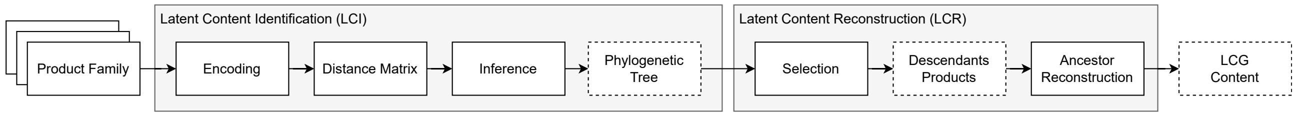


Figure 2: Our Latent Content Generation Approach.

parts: Latent Content Identification and Latent Content Reconstruction. First, the products in the family are fed into the phylogenetic algorithm to generate the phylogenetic tree. First, each product is encoded into genetic information and then analyzed by pairs to create the distance matrix. To finally create the phylogenetic tree, we perform phylogenetic inference on the collection of taxa, resulting in the visual diagram. Second, With the tree, the developer can select two genetically related products (descendant products) and perform the Latent Content Reconstruction operation to raise the common ancestor between the two products leveraging the information on the commonalities and differences of each descendant.

This approach serves as the first step in the formalization of variability. The generated tree provides an ordered and structured view of the family. We can also create new content that is ensured to be within the family scope by reconstructing common ancestors.

3.1 Encoding

The first step in the phylogenetic analysis is to encode the content into genetic information. Video game development, a wide range of game content items (e.g., levels/stages or characters) are defined as software models using commercial tools for visual scripting in engines such as Unreal (Blueprints) or Unity (Unity Visual Scripting). A recent survey shows that it is also common to apply Domain Specific Languages (DSLs) [57] for this task. The original DSL used to specify NPCs in Kromaia is SDML. In this work, we apply a string encoding to represent the software models, as others have done for years [8]. It is done by transforming the SDML model into a single string of characters called a genome. Each NPC element (hull, link, weapon, etc.) is represented as a single character in the string. It works similarly to a regular DNA sequence where the order and value of each character are relevant to encode the data.

3.2 Distance Matrix

When comparing two products, the resulting distance value is stored in a square matrix called distance matrix, as it is commonly used in biology [53]. The values are extracted from 0 (exact copy) to 1 (each gene of the genome is different).

To calculate this distance, we compare each individual's genome with the other individuals using the Levenshtein distance algorithm [31]. The larger this distance, the smaller the chance that a model is evolved into another model by a series of changes. However, other distances can be used or created in the future. Each gen is compared one by one, and then the average is computed (i.e., the gene ABA compared with the gene ABC would compute a 0.333).

3.3 Phylogenetic Tree Inference

We use the Neighbor-Joining Method from the distance matrix as an inference technique to produce a phylogenetic tree. This method is commonly used in biology phylogenetics [46]. From the genetic distance matrix, each element is compared from the most similar (closer to 0) to the most different (closer to 1). Each comparison creates a new element that replaces the compared elements. These new elements are compared with the other elements using the average of the original values. Here is an example to illustrate the inference calculation. The genetic matrix has the following tuples: A|B = 0.5, A|C = 0.25, B|C = 0.33. The A|C tuple will be processed by replacing A and C with AˆC. Then, the tuple AˆC|B is processed with the average of the original values: AˆC|B = (A|B+C|B)/2 = 0,415.

This process is repeated for each pair, storing the creation order to build the tree until only two elements are linked by the root. This root will store the highest value, representing the common genes in all genomes. Figure 1 shows an example of a simple phylogenetic inferred from a distance matrix. In that case, the distance $d_1 + d_2 + d_3 + d_4$ should be similar to the distance included in the matrix.

This tree can serve as a visual representation of the product family used as input and the relations among products in terms of genetic similitude. The game developer can spot possible gaps that can be exploited to create new content. These gaps are known in the phylogenetic realm as common ancestors. These ancestors are not present in the original family and serve as hypothetical individuals from which the current individuals could have evolved. We leverage the ancestral data to identify novel hypothetical entities called latent content. While not previously documented, these entities are not entirely unprecedented, embodying characteristics inferred from their descendants.

By inferring the characteristics of newly created individuals from their descendants, we maintain the characteristics of these individuals aligned with the family scope, whether this scope is formalized or not. As video game content is creative by nature, the family scope can sometimes be determined by subjective human requirements such as visual appeal or the fun of the game. Thus, the scope of a family of video game content is not always formalized.

3.4 Latent Content Reconstruction

In the phylogenetic tree used in our work, the individuals provided as starting points act as leaves, while the inner, inferred nodes represent points of inflection concerning lineage. When the sibling of a taxon (leaf) is a taxon too and not an inner node, the parent inner node, which is their Most Recent Common Ancestor (MRCA), hypothesizes a taxon from which those two existent ones descend [10]. Our approach uses two taxa with a shared MRCA to reconstruct its genetic material to produce an individual that realizes the latent game content represented by that MRCA.

The reconstruction of that ancestor is done manually, following a set of rules that we defined after studying the information retrieved in interviews with the developers of the video game case study. These criteria take into account the concepts of age and antecedent for elements like files or weapon/projectile types, referring to the

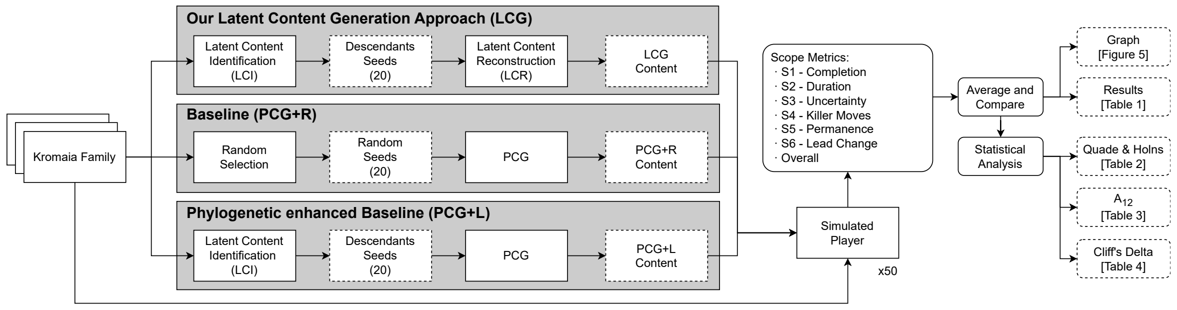


Figure 3: Experimental Setup

date when they were implemented by the developers or the type that could be considered as the antecedent of the other two given types. The reconstruction rules can be summarized as follows:

- **Anatomy:** The anatomical structure keeps the base infrastructure of the biggest of the two descendants, in terms of spatial organization, but the amount of hull is the average. Additionally, the links that need to be broken due to that new, average organization are removed accordingly.
- **Visual Appearance :** In the case of anatomical characteristics shared by the two descendants, which only differ in the mesh file that gives an object its visual appearance, the oldest of the two visual appearances is kept in the reconstructed ancestor.
- **Weaponry:** The weapon amount is the average of the two descendants, while the weapon types assigned are those considered to be the antecedents of the two when they differ. This criterion is applied to projectile types, too. In both cases, the parameters used could be set by choosing randomly one of the following options: The complete parameter set from the oldest of the two weapon/projectile types and averaged values for every parameter.
- **Behavior:** In the same manner as weaponry, a random decision is made with regard to the use of antecedent types or average parameters.
- **Cosmetic Content:** The objects embedded in the anatomy with purely cosmetic purposes and showing, optionally, animations are only kept if both descendants share them. Otherwise, the reconstructed ancestor will not include those parts.
- **General parameters:** The parameters dealing with characteristics like scale or time values present globally in the descendants in the different aspects described previously are averaged.

These rules represent the first step in consolidating a specific Latent content reconstruction operation. Our purpose is to explore, in the future, possible further installments of that operation, taking into account new perspectives.

4 EVALUATION

To evaluate our approach, we have designed an evaluation to address the following research questions:

- **RQ1** - Does our Latent Content Generation approach produce results that fit the scope of the family in terms of the metrics used for families of video game content better than the state-of-the-art approach for PCG?
- **RQ2** - If so, how big are the differences between both approaches?
- **RQ3** - Does the Latent Content Identification operation produce better seeds than random for the state-of-the-art approach for PCG?

The state-of-the-art PCG approach is presented in Section 4.3.

4.1 Kromaia Product Family

The main artifact in our evaluation is video game content. Specifically, it is content from Kromaia[1], a video game released on both PC and console. This game is distinguished by its dynamic 360-degree movement flight system and shooting mechanics.

The game features Non-Playable Characters (NPCs) that players must overcome. These NPCs vary widely, ranging from small, stationary enemies to large, articulated ones with complex behaviors and weapons. All NPCs align with the developer's vision, adhering to consistent mechanics, dynamics, and aesthetics [26].

During the development of the game, these NPCs were created using opportunistic reuse by manually copying and pasting parts of some NPCs already created. This opportunistic approach created a product family that has similarities among NPCs. There are common parts that were borrowed from other NPCs, and there are new parts that are particular and unique to each NPC.

These NPCs incorporate concepts common to all entities in the game: hulls, which are rigid bodies in the character's anatomy, such as beads on a necklace or body parts of a creature; links representing joints with varied degrees of freedom that connect hull pairs; weak points added to regular hulls to define damageable objects; weapons used to attack other characters via projectiles or spikes; and AI components that parametrize behaviors.

4.2 Experimental Setup

Figure 3 shows the experimental design of the evaluation. The left part shows the family of Kromaia video game content used as input for the evaluation. The family is composed of 48 enemies present in the commercial release of Kromaia. Then, this family is fed as

[1]https://store.steampowered.com/app/285980/Kromaia/

input to the three content generation approaches compared: our Latent Content Generation approach, the state-of-the-art approach for PCG used as a baseline, and a version of the baseline enhanced using our Phylogenetic operation (see grey boxes in Figure 3).

Then, the content generated by the three approaches is evaluated using video game simulations and a set of metrics used by the GSE community to assess video game content and guarantee scope alignment (*Completion, Duration, Uncertainty, Killer Moves, Permanence, Lead Change*) [8]. In addition, the original content used in the evaluation is also assessed using those metrics (see dashed arrow). To compare the new content produced with the original content in terms of those metrics to determine if the new content generated is within the scope of the existing content.

Finally, the data is processed to present the results of the evaluation (see right part of Figure 3). First, differences among the scope metrics of the new content generated and the original content are computed and presented. Then, a statistical analysis is performed, including measures of statistical significance (Quade and Holm's) [19] and effect size ($\hat{A}_{12}$ and Cliff's Delta) [14, 15, 51].

4.3 Content Generation Approaches

To answer the research questions, we compare our LCG approach with the baseline PCG. To determine the PCG Baseline, we had to identify work from PCG literature capable of generating game elements such as the ones of Kromaia. After surveying the literature, we chose the work by Gallota et al. [18] because (1) it is one of the most representative search-based PCG approaches; (2) it generates spaceships for the Space Engineers videogame, so it seemed capable of generating content for Kromaia, and (3) it achieves the best results for this kind of content [18]. The approach from Gallota et al. is a hybrid Evolutionary Algorithm, combining an L-system with a Feasible Infeasible Two Population Evolutionary Algorithm.

Additionally, we explore another way of generating new content, which is the result of combining the operation for selecting the seeds via phylogenetics (the ancestor identification operation) and the PCG evolutionary algorithm. We perform this combination to see how each component affects the results, i.e., we want to observe how the Latent Content Identification operation can affect the PCG algorithm and, thus, the content generated and how the Latent Content Reconstruction operation affects the results.

As a result, there are three sets of new content generated (see the middle part of Figure 3):

- **LCG**: Content created by our Latent Content Generation approach. That is the combination of the seeds obtained with our Ancestor Identification operation followed by the selection of descendants using the phylogenetic tree fed to our ancestor reconstruction operation.
- **PCG+R**: Content created by the baseline PCG approach, fed with random seeds (the usual practice in the literature).
- **PCG+L**: Content created by the baseline PCG approach, fed with the descendant seeds identified by our Ancestor Identification operation, followed by a selection of descendants using the phylogenetic tree.

4.4 Game Simulations

The generated content is then evaluated using video game simulations, an accepted practice [11] for video games. Simulations are easy to find in video games, as many of the elements of the game are autonomous (NPCs), and other games directly include contenders (such as racing games).

For the evaluation, we use a simulated player that performs duels against the enemies, simulating what a human player would do during a duel and generating tracing data that will be used to calculate the scope metrics. In the case of Kromaia, the simulated player has been created by the development team, including the configurations needed for their specific intention for the game.

The simulated player is used in our evaluation to execute automatic duels against any type of enemy (the original Kromaia enemies or the generated ones). In those simulations, the simulated player confronts the enemy, strategically moving and targeting the enemy's different hulls and weak points to destroy them. Meanwhile, the enemy will act based on its anatomical structure, behavioral patterns, and attack/defensive dynamics, aiming to destroy the simulated player. Both entities actively strive to emerge victorious, avoiding draws or ties and ensuring a definitive win. As the simulated player is non-deterministic, we will run each duel 50 times to ensure the consistency of the results across executions. It is no more than 50 times due to time execution constraints. However, it is within the suggested range in the literature [3].

4.5 Scope Metrics

Video games are complex software because they interconnect work from various roles (game designers, programmers, 3D artists, or musicians) into a single piece transmitting a pleasant or fun experience to the player. However, 'fun' is an abstract concept and depends on the game designers' intention for their audience. What is fun for some (e.g., a horror movie) can be unpleasant for others. The game designer's mission is to make decisions, take necessary risks, and ensure all elements fit the intended game scope.

Regarding the content of the video game, it must be properly aligned with the experience being evoked to empower game designers to effectively convey their intended experience [48]; mechanics, dynamics, and aesthetics of a video game are interrelated [24], aligned content avoids disrupting the player experience. Therefore, when generating new content for an existing video game (e.g., the family of content from Kromaia), we must ensure that the new content is aligned and remains within the same scope.

There are measurable indicators of game quality in the literature. In this work, we will apply six widely accepted indicators that fit the type of game used in our evaluation (Kromaia, a 3D spaceship shooter) [8, 11]. Our evaluation measures the six scope criteria from 0 to 1 (values will be clamped when needed). Then, we calculate the distance between the generated content and the family of content from Kromaia, obtaining low values for content that is effectively aligned with the scope of the family.

S1-Completion: A duel against an enemy should end with more conclusions (victories for any of the two contenders) than ties (no contender wins, and the duel keeps going for longer than expected by the game designers). Depending on the type of game, duels can be designed to last longer or shorter, so the timeout to consider a duel as a tie can be adapted. In this type of game, ties are abnormal and usually indicate a problem with the content (e.g., cannot be killed by any means). The criterion S1-Completion is the ratio of conclusions over total duels:

$$\text{S1-Completion} = \frac{Conclusions}{Duels} \tag{1}$$

S2-Duration: The duration of duels between players and enemies is expected to be around an optimal value ($T_{Optimal}$) stated by the game designer. Significant deviations from that reference value are good design-flaw indicators: short games are probably too easy, and long duels tend to make players lose interest. The criterion S2-Duration is the average difference between the duration of each duel (T_d) and the optimal duration ($T_{Optimal}$):

$$\text{S2-Duration} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels}\right) \tag{2}$$

S3-Uncertainty: If a duel outcome can be foreseen in advance, the player might lose interest when she realizes that she will lose the duel and will be bored for the remaining time. To keep players engaged during the duel, the contenders should not get extremely close to victory or defeat too early before the duel finishes. Therefore, a duel is considered to be more uncertain the longer the time until the player's or the enemy's health levels are too low, considered as a dangerous value (P_d and B_d, respectively). For each duel, S3-Uncertainty measures the average deviation between the time at which one of the contenders reaches a dangerous health value and the total duration of the duel (T_d).

$$\text{S3-Uncertainty} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min(P_d, B_d)}{T_d}}{Duels}\right) \tag{3}$$

S4-Killer Moves: While in a duel, contenders perform actions (e.g., moving closer or farther, shooting, using special weapons). Some actions are considered trivial, and others are considered a remarkable highlight (H) towards the duel's outcome. In contrast, others are considered Killer moves (K) as they truly determine the duel's outcome. S4-Killer Moves measures the ratio between killer moves (K) and remarkable highlights (H).

$$\text{S4-Killer Moves} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels}\right) \tag{4}$$

S5-Permanence: Permanence measures how often the advantages given by significant actions or moves by one of the contenders are immediately reverted by the opponent in terms of dominance. Recovery moves (R) are those that quickly cancel the advantages gained by the opponent by a killer or highlight move. The criterion S5-Permanence is measured as follows:

$$\text{S5-Permanence} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d + K_d}}{Duels}\right) \tag{5}$$

S6-Lead Change: The lead of a duel is determined at any given moment by considering the contender with the highest health level. S6-Lead Change measures how often a highlight or killer move changes the lead of the duel (L) as the ratio between changes in the lead (L) and the number of highlight or killer moves (H,K) during the duel:

$$\text{S6-Lead Change} = clamp_{[0,1]}\left(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d + K_d}}{Duels}\right) \tag{6}$$

Overall Finally, the six scope criteria are combined into a single average value representing the overall scope differences of the evaluated content. This will ease the interpretation of the results:

$$\text{Overall} = \frac{\sum_{i=1}^{N} S_i}{N} \tag{7}$$

4.6 Implementation Details

The evaluation has been performed using a commercial HP laptop with an Intel i7-10750H processor, 16Gb of RAM, and Windows 10 64 bits as the host operating system. The execution of the phylogenetic phase took around 10 minutes, with the calculation of the genetic distance matrix being the most time-consuming part (95% of the execution time). This bottleneck happens due to the expensive distance calculation as it needs to iterate for each character of the NPC files. The presented approach for Latent Content Generation has been implemented using the .NET 6 runtime environment and C# 10 as the programming language.

The scope metrics include configurable parameters tailored to the needs of the game designers. In this work, parameters for those metrics have been provided by the Kromaia development team (determined based on their own experience, desires, or through questionnaires with players) and are as follows: S1-Completion, 20 min. is the maximum time for a duel; S2-Duration, the optimal time for a duel, is 10 min.; S4-Killer Moves, a highlight move happens when either the boss unit or the player experiences a decrease in health, and killer moves are those that make the difference in health between the contenders reach 30%.

The statistical analysis has been performed using R-Studio. The results of the evaluation, the implementation of the approach, the game simulations, and the scripts used for the statistical analysis are made publicly available to the reader[2].

4.7 Results

This section presents the results obtained after applying the content generation approaches.

The phylogenetic tree generated from the Kromaia product family is presented in Figure 4. This tree has a structured view of the

[2]https://anonymous.4open.science/r/LatentContentGeneration-4F32

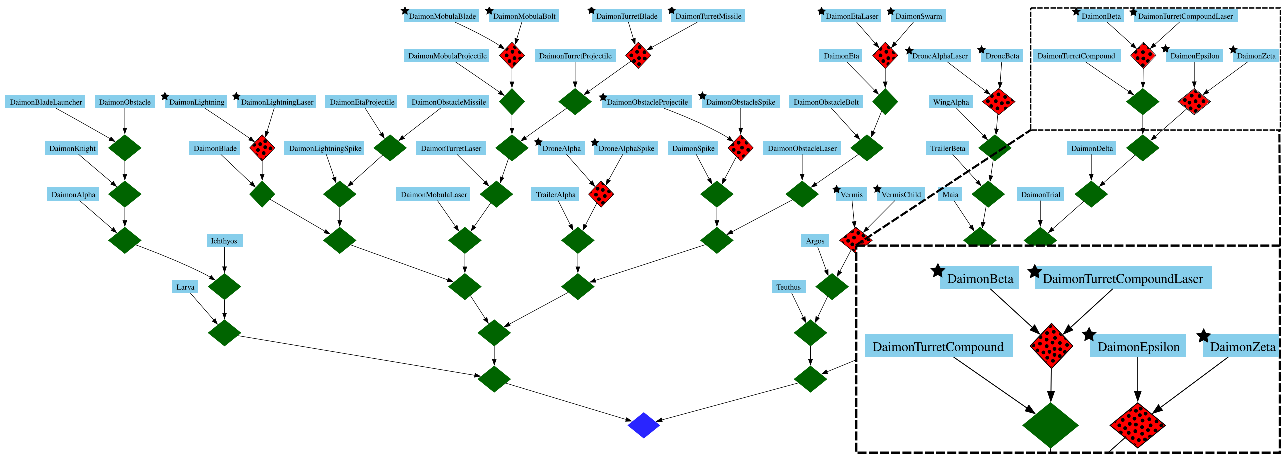


Figure 4: Phylogenetic Tree of the family of enemies from Kromaia. Each enemy used as input is depicted as a blue rectangle; ancestors identified are depicted as green diamonds; ancestors reconstructed are depicted as red dotted diamonds.

	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
Scope	0.99997 ± 0.00015	0.18663 ± 0.01565	0.01313 ± 0.00754	0.82573 ± 0.00232	0.9929 ± 0.0021	0.16053 ± 0.00412	0.52982 ± 0.00399
LCG	0.00003 ± 0	0.07369 ± 0.05910	0.01340 ± 0.00449	0.05175 ± 0.03893	0.00790 ± 0.00881	0.03169 ± 0.4389	0.01122 ± 0.01077
PCG+L	0.00003 ± 0	0.13263 ± 0.06105	0.01792 ± 0.01856	0.14479 ± 0.11813	0.00941 ± 0.00905	0.08366 ± 0.06686	0.02865 ± 0.02102
PCG+R	0.00003 ± 0	0.14780 ± 0.05186	0.02005 ± 0.02252	0.14359 ± 0.12938	0.00954 ± 0.00820	0.08178 ± 0.07408	0.02989 ± 0.02057

Table 1: Reference values for each scope metric and differences for each content generation approach and scope metric. The lower the values the better. The smallest value for each scope metric is highlighted in grey.

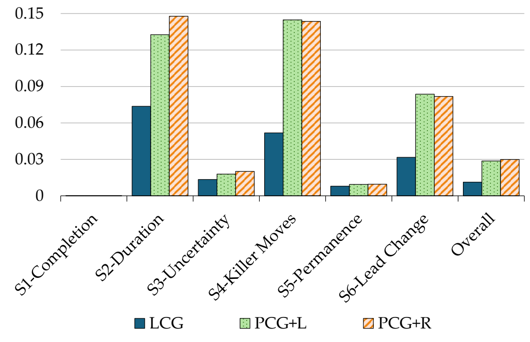


Figure 5: Scope differences between the content generated and the family of products from Kromaia. The lower the differences, the better.

product's family and the genetic relations between products. It serves as the starting point for the reconstruction of latent content. The phylogenetic tree provides descendants from the tree's leaves (blue rectangles) and its ancestor to be reconstructed (green diamonds). Each ancestor used in the evaluation is represented as a red dotted diamond, and the root of the tree, the common ancestor to all NPCs, is shaded in blue. The selection of ancestry was performed by one professional video game developer of Kromaia with 20 years of experience, and his selection was corroborated by another Kromaia developer. The bottom-right part of Figure 4 shows a zoom of two latent ancestors and their corresponding descendants.

As an example, we will use the reconstructed ancestor of the enemies DaimonMobulaBlade and DaimonMobulaBolt. Both enemies have similar anatomy, visual appearance, and behavior. The main difference is the weaponry. Then, the reconstructed ancestor has the common parts and a random weapon from each enemy with averaged parameters.

The six metrics are obtained by simulations executed for each NPC created and for each product in the Kromaia family. The results show that with LCG, we can create new content more similar to the NPCs in video games. Figure 5 and Table 1 show the differences for each scope metric between the content generated and the Kromaia family for each approach evaluated. The lower, the better. The differences between our LCG approach and the reference scope metrics from the family (see the first row of Table 1) are smaller than those created by the baselines (PCG+R, PCG+L).

We can see that the S1-Completion metric is identical to the scope in the three approaches measured. S1-Completion is the minimum requirement an enemy should meet, for it is expected that the battles, at least, can be completed. Every approach can create content with basic functionality as well as those present in the family. S3-Uncertainty and S5-Permanence are the two other metrics that are more similar to the scope. With these metrics, we can see how LCG starts to outperform the baseline, but the differences seem negligible.

S2-Duration, S4-Killer Moves, and S6-Lead Change are the metrics that show the most difference from the scope. The S2-Duration for LCG is similar to the optimal duration, as indicated by the game designer. S4-Killer Moves and S6-Lead Change show similar performance among approaches, as was expected because these metrics are related with almost three times more in scope in favor of LCG.

All these metrics are summarized with the Overall metric and show that, with LCG, developers can create content that is more similar to the scope than PCG, which is more than twice as different.

4.8 Statistical Analysis

To address RQ2, we compare the results obtained from the different content generation approaches. All the data obtained from the simulations were compared to the scope of the family of products and then analyzed following established guidelines [3].

First, a statistical significance analysis will provide formal and quantitative evidence of the differences among approaches, determining whether the differences observed are due to the application of different generation approaches or mere chance. Then, an effect size analysis will be performed to determine if those differences are significant in practice.

4.8.1 Statistical Significance. To determine the statistical significance, we defined two hypotheses: H_0 is the null hypothesis, which states that there are no differences among the content generation approaches; H_1 is the alternative hypothesis, which states that the results of at least one content generation approach differ from another.

Then, a statistical test that returns a probability value (p-value) in the range of 0 to 1 is run. The p-value indicates the probability of the null hypothesis being false (and thus H_0 is rejected, and the alternative hypothesis H_1 is accepted). In this field, a p-value under 0.05 is considered statistically significant.

The statistical test to be applied depends on the nature of the data; in this case, our data does not follow a normal distribution and then requires a non-parametric test. We chose the Quade test, which has shown more power than the rest when applied to a low number of approaches (under 5) and using real data [19].

The first row of Table 2 shows the results for the Quade test applied to each scope metric separately. The p-value is below the 0.05 threshold for all metrics except S1-Completion, resulting in an inconclusive test (expected as all values were 0 or almost 0). Therefore, we conclude there are no differences in the first metric (S1-Completion) and differences due to the use of different content generation approaches for the rest of the metrics.

However, the Quade test is only valid to conclude that there are differences but is not able to point out which content generation strategy is better. We need to compare the results of each pair of approaches separately. Holm's post hoc procedure does this, another statistical test commonly used in conjunction with Quade [19]. The test is run for each metric separately, omitting the S1 metric as Quade already showed that differences were not significant.

The second to the fourth row of Table 2 shows the p-values of Holm's Post Hoc for each quality metric (columns) and each pair of generation approaches (rows). Values below the threshold (0.05) are highlighted in grey, indicating that the difference in the results for that pair of approaches and scope metric is significant.

We can see that the differences between LCG and the two PCG approaches are significant for the six metrics analyzed in this test (S2, S3, S4, S5, S6, and Overall). According to Holm's test, the differences between both PCG approaches, with latent input or with random input, are not significant except for S3 - Uncertainty.

4.8.2 Effect Size. After concluding that there are differences among the results of the different content generation approaches, we have to determine the effect size of those differences, that is, how big those differences are [21]. Even when having differences, they can be too small and have no practical value (especially when the number of runs is big enough). Therefore, we assess the magnitude of that difference by applying two non-parametric measures, Vargha and Delaney's $\hat{A}_{12}$ [51] and Cliff's Delta [14, 15].

Table 3 shows the $\hat{A}_{12}$ statistic for each scope metric (columns) and pair of generation approaches (rows). For example, the first row, the third column in Table 3 shows the $\hat{A}_{12}$ value for LCG vs PCG+L for the S2-Duration metric, 23.40%. It indicates the probability that an observation from the first group (LCG) is greater than an observation from the second group (PCG+L). In this case, as the data represents scope differences, the lower the difference, the better. That is, in 23.40% of the runs, the differences in scope between the LCG results and the reference values in terms of the S2 metric are bigger (or worse in this case) than the corresponding differences of the PCG+L approach. Similarly, the opposite is also valid, so in 76.6% of the runs, the differences in the scope of LCG are smaller than the differences of PCG+R for the S2 metric.

Table 4 shows the Cliff's Delta statistic for each scope metric (columns) and pair of generation approaches (rows). It indicates the degree to which two distributions overlap. Negative values indicate that values from the first approach are smaller than those from the second. In this case, the lower the value, the better for the first approach (as the differences in the scope reference are smaller).

For both $\hat{A}_{12}$ and Cliff's Delta, we observe again that S2-Duration, S4-Killer Moves, and S6-Lead Change are the metrics that present the biggest differences between LCG and the PCG baseline. The other metrics perform similarly for the three approaches. Based on the Cliff's Delta statistic, we can conclude that LCG can create content that is a better fit for the family's scope than PCG+L (with a medium effect size) and PCG+R (with a large effect size).

5 ANSWER TO RQS AND DISCUSSION

As an answer to RQ1, we can conclude that our LCG approach can produce results that fit the scope of the family of products in terms of the metrics studied better than a state-of-the-art approach. In particular, the differences in scope between the family of products and the content generated by our LCG approach are smaller than the corresponding differences for the baselines for all the scope metrics studied (see Table 1).

As an answer to RQ2, the results for our LCG approach are better than those produced by a state-of-the-art approach, generating individuals with smaller scope differences, 75.88% of the runs for the overall scope metric according to the $\hat{A}_{12}$ effect size (see Table 3). This can be interpreted as a medium/large difference according to the Cliff's Delta effect size (see Table 4).

		S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
Quade	All	-	$\ll 2x10^{-16}$	$3.0x10^{-13}$	$\ll 2x10^{-16}$	$7.0x10^{-8}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
Holm's	LCG vs PCG+L	-	$\ll 2x10^{-16}$	0.00014	$\ll 2x10^{-16}$	0.02	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	LCG vs PCG+R	-	$\ll 2x10^{-16}$	$1.1x10^{-9}$	$\ll 2x10^{-16}$	0.02	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	PCG+L vs PCG+R	-	0.12	0.01928	0.059	0.99	0.63	0.36

Table 2: Quade test (first row) and Holm's Post Hoc (second to fourth rows) p-values. Values below 0.05 are highlighted in grey

$\hat{A}_{12}$	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
LCG vs PCG+L	50%	23.40%	45.58%	26.70%	44.91%	25.76%	26.55%
LCG vs PCG+R	50%	18.31%	42.74	31.10%	45.47%	27.43%	24.12%
PCG+L vs PCG+R	50%	44.62%	47.26%	51.98%	50.03%	51.09%	47.55%

Table 3: $\hat{A}_{12}$ statistic. The lower the values the better for the first approach compared. Values below 30% are highlighted in grey.

Cliff's Delta	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
LCG vs PCG+L	0 (negligible)	-0.532072 (large)	-0.088312 (negligible)	-0.465984 (medium)	-0.10184 (negligible)	-0.484864 (large)	-0.468916 (medium)
LCG vs PCG+R	0 (negligible)	-0.633744 (large)	-0.14512 (negligible)	-0.378092 (medium)	-0.090664 (negligible)	-0.451472 (medium)	-0.517696 (large)
PCG+L vs PCG+R	0 (negligible)	-0.10768 (negligible)	-0.054824 (negligible)	0.0396 (negligible)	0.000544 (negligible)	0.021892 (negligible)	-0.04898 (negligible)

Table 4: Cliff's Delta statistic. Medium and large effect sizes are highlighted in grey.

As an answer to RQ3, when using the seeds obtained by our Ancestor Identification operation to generate content with a state-of-the-art approach, the differences in scope are smaller than when using randomly selected seeds. However, those differences are small and only produce better scope values 52.45% of the runs.

We can explore video game content from a new perspective by mixing the two concepts of Phylogenetics from the biology domain and Product Family from the Software Product Line domain. We can leverage the already-created content with its information and knowledge to discover new content that was already there waiting to be discovered. This latent content is particularly interesting as it is ensured to follow the present design guidelines whether they are formalized or not; the content is within the family scope.

Our results demonstrate that content created with LCG is more similar to the already present content. This means the content is nearer to the production-ready state for quick incorporation into the final game. The content created via LCG also has a higher chance of following the creative intentions of the game designers.

Additionally, the structured view that the tree provides can help developers understand the product family's relations and scope thus easing software variability. This can help developers make creative decisions on how the game should be structured in terms of content variability. Designers can choose genetically near NPCs to place them at the same level as they are more similar and, thus, cohesive with the general intent of said level. Or if the designer wants a wide variety of NPCs for creative (narrative, aesthetics, mechanics, etc.) reasons, the phylogenetic tree provides that information.

Video games have large amounts of content and the phylogenetic tree can be a good way to introduce SPLs into video games that are already advanced in development. The fact that developers can see genetic relations among content can derive into feature models that conform to the discovered relations, similar to the approach of König et al. [27] called taxonomy mining.

Finally, referring to the simple phylogenetic tree in Figure 1: Would the common ancestor between the owl and the gecko be a valid product? Our results say yes, but more importantly, LCG invites developers to explore that specific content as it may produce a better individual within the video game's scope.

6 THREATS TO VALIDITY

To address the limitations of our evaluation, we adopt the validity threat classification framework from Wohlin et al. [55], which identifies four dimensions of validity threats:

Construct Validity: This dimension examines whether the operational measures accurately reflect the true theoretical constructs they aim to represent. We addressed this threat by performing a fair comparison between our approach and the baseline using the same metrics and the same simulated player. Additionally, the six metrics used in the evaluation are accepted by the game software engineering community [8, 11].

Internal Validity: This dimension is concerned with the causality relationships established within the study. It evaluates whether the outcomes can genuinely be attributed to the treatment or intervention being researched rather than being affected by other variables. The results could be affected by implementation details; to mitigate this, we created ten different new products with each approach and executed the simulated player 50 times for each, accounting for random variation. Additionally, we have provided the implementation details and the source code.

External Validity: This dimension pertains to the generalizability of the study results to other settings, environments, or groups. The phylogenetic encoding generalizes the implementation, and thus, the distance calculation and inference are agnostic to the domain. The genetic relations present in the phylogenetic tree do not represent an ad hoc implementation for the video game content domain, as that content was genetically encoded. Our evaluation was performed over a single industry-scale video game, mitigating the lack of real problem instances. However, our findings should be replicated using other video games and alternative content to confirm the generalizability properly.

Conclusion Validity: This dimension focuses on the reliability and accuracy of the conclusions drawn from the study. It involves ensuring that the conclusions about relationships or differences are statistically and methodologically sound. We provided ten individuals for each approach comparison and executed the simulation 50 times for each individual. We also performed a statistical analysis that is widely accepted in software engineering [3] including statistical significance (Quade test and Holm's Post Hoc) and effect size (Cliff's Delta and $\hat{A}_{12}$) statistics.

Finally, one limitation of this approach is that it needs an existing set of products to create the phylogenetic tree. This is not the case for other PCG approaches. Our approach can render better results for specific stages of game development where there is already content created.

7 RELATED WORK

This section presents the related works taking into account the terms SPL in Video Games, Content Generation in Video Games, and Phylogenetics in Software. We can see that while there are works that tackle these concepts or similar ones, none of them use phylogenetics to generate content through the lens of SPL in an industry-scale case. Our research aims to create video game content that is not only novel but also in line with the developers' vision.

7.1 SPLs in Video Games

Some papers address the variability of video games' content and software. Some approaches are focused on creating engines that help game developers make games more scalable, independent, and reusable. Such is the case of the Minimal Engine for Digital Games (MEnDiGa) created by Boaventura and Sarinho [9], an extension of their previous engine called FEnDiGa [47]. In MEnDiGa, they refined the development of logic features and modules that represent and adapt game features, enabling functionality across multiple gaming platforms. Additionally, Castro and Werner [12] discuss a game prototype developed using a dynamic SPL to generate game modifications systematically. This prototype showcases the feasibility of automating the modification process, positioning the original game as the central component of the game's functionalities.

Additional research efforts delve into developing Software Product Lines (SPLs) through re-engineering processes. Lima et al. [32, 34] introduce two studies concerning the recovery of Product Line Architecture (PLA). They implemented their previously proposed guidelines [33] to establish the PLA for the Apo-Games project [28]. Moreira et al. [41] examine empirical data from the re-engineering efforts of two open-source projects, ArgoUML and Phaser. Their findings reveal significant differences in the re-engineering processes between ArgoUML-SPL and Phaser. Common challenges are encountered in both projects, including a scarcity of tools, resulting in incomplete and inconsistent feature extractions, complexities in managing feature dependencies with a compositional approach, and the absence of a variability model to address feature constraints.

Similarly, Martinez et al. [37] share insights from their experience in creating a Software Product Line (SPL) through re-engineering system variants centered around an educational game called Robocode. They explore their findings from various angles, including the educational value, the extraction process, and the time and effort involved. Debbiche et al. [16] investigate the Apo-Games to pinpoint reusable code or artifacts. Their analysis covered five Java games, with three of these games subsequently transitioned into a composition-based SPL using *FeatureHouse* [2]. They claim that maintaining the testability of the SPL ensures accurate code transformations and recommend the incremental incorporation of new features to ease the extraction process.

7.2 Procedural Content Generation from Game Software Engineering

Relevant to PCG, Preuss et al. [45] examine the interplay between quality and diversity in PCG for game development. Their research involved an experimental evaluation of various algorithms and distance measures using a tool designed for generating game levels. They concluded that the Niching Evolutionary Algorithm 2 (NEA2) effectively balances quality and diversity, contingent upon the employment of a robust distance function. Subsequently, Gravina et al. [20] characterized quality diversity as a pivotal search strategy within search-based PCG.

The study conducted by Melotti et al. [40] introduces and implements the Deluged Novelty Search Local Competition algorithm (D-NSLC), which utilizes morphological niches to promote solution diversity in PCG for a game. D-NSLC segments the population into distinct niches and targets the optimal individuals within each while exploring the search space. They conducted an experiment within a roguelike video game context using four different setups. The findings underscored the advantages of the Novelty Search, notably its significant contribution to generating diverse individuals.

Finally, Blasco et al. explored the application of Search-Based Software Engineering (SBSE) for content generation [7, 8]. These studies utilized an evolutionary algorithm steered by simulations that incorporate the generated content. They use an evolutionary algorithm known as Evolutionary Model Generation (EMoGen) to produce software models quickly and efficiently. The research demonstrated that models generated by EMoGen for the commercial video game Kromaia were comparable in quality to those created manually by developers but took significantly less time.

None of the aforementioned studies leverage the idea of latent content with genetic divergence points and ancestry branches. This work opens a new path where the focus is that the content is aligned with the scope precisely with the bottom-up relationship. In the video game domain, the scope is crucial because the developers' vision for the game is critical and difficult to formalize.

7.3 Phylogenetics in Software

To the best of our knowledge, there is no study that applies phylogenetics to software. However, the study by König et al. [27] focuses on enhancing the understanding and application of software variability through the technique of taxonomy mining. The authors present a methodological approach aimed at extracting and utilizing taxonomies to manage variability in software product lines better. Their approach is applied to generate taxonomy graphs of different SPLs in Software Engineering. Taxonomy and Phylogeny are closely related sciences in the realm of biology. Taxonomy focuses on the study of the classification of species, and Phylogeny focuses on the study of evolutionary relationships between organisms. They have different objectives using the same information.

8 CONCLUSION

Our LCG approach proves itself to be a valid approach for content creation in video games. The evaluation has demonstrated its capabilities to identify gaps within a product family, hidden content waiting to be discovered. The content created via LCG is more similar to the current scope of an industry-scale video game.

Phylogenetics can manage variability by pointing out genetic relations among individuals of the same product family. Identifying these relations can be key for the future of software variability, providing developers with a full view of an SPL at a glance. This can enhance strategic decision-making regarding the development and deployment of additional products.

Specifically, in video games, the phylogenetic tree can help designers arrange and manage their content while guiding the next steps in the creation of new content. Also, our approach can potentially be used in other domains beyond video game content creation. Genetic divergence points and ancestry branches can provide new perspectives to variability thanks to the phylogenetic approach.

ACKNOWLEDGMENTS

This work has been partially supported by the following: Spanish National R+D+i Plan and ERDF funds under the Projects VARNETICA (CNS2023-145422) and VARIATIVA (PID2021-128695OB-I00); Centro para el Desarrollo Tecnológico y la Innovación (CDTI) (SAV-20221011); Gobierno de Aragón (Spain) (Research Group S05 20D); And by FEDER/Ministry of Science, Innovation and Universities/Junta de Andalucía/State Research Agency/CDTI with the following grants: TASOVA PLUS research network (RED2022-134337-T).

REFERENCES

[1] Apostolos Ampatzoglou and Ioannis Stamelos. 2010. Software engineering research for computer games: A systematic review. *Information and Software Technology* 52, 9 (2010), 888 – 901. https://doi.org/10.1016/j.infsof.2010.05.004
[2] Sven Apel, Christian Kästner, and Christian Lengauer. 2013. Language-Independent and Automated Software Composition: The FeatureHouse Experience. *IEEE Transactions on Software Engineering* 39, 1 (2013), 63–79. https://doi.org/10.1109/TSE.2011.120
[3] Andrea Arcuri and Lionel Briand. 2014. A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering. *Softw. Test. Verif. Reliab.* 24, 3 (May 2014), 219–250. https://doi.org/10.1002/stvr.1486
[4] Earl T. Barr, Yuriy Brun, Premkumar Devanbu, Mark Harman, and Federica Sarro. 2014. The plastic surgery hypothesis. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 306–317. https://doi.org/10.1145/2635868.2635898
[5] D.A. Baum and S.D. Smith. 2012. *Tree Thinking: An Introduction to Phylogenetic Biology.* Macmillan Learning. https://books.google.es/books?id=zW_ApwAACAAJ
[6] T. Biggerstaff and C. Richter. 1987. Reusability Framework, Assessment, and Directions. *IEEE Software* 4, 2 (1987), 41–49. https://doi.org/10.1109/MS.1987.230095
[7] Daniel Blasco, Jaime Font, Francisca Pérez, and Carlos Cetina. 2023. Procedural content improvement of game bosses with an evolutionary algorithm. *Multimedia Tools and Applications* 82, 7 (2023), 10277–10309.
[8] Daniel Blasco, Jaime Font, Mar Zamorano, and Carlos Cetina. 2021. An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering. *Journal of Systems and Software* 171 (2021), 110804.
[9] Filipe M. B. Boaventura and Victor Travassos Sarinho. 2017. MEnDiGa: A Minimal Engine for Digital Games. *Int. J. Comput. Games Technol.* 2017 (2017), 9626710:1–9626710:13. https://doi.org/10.1155/2017/9626710
[10] Andrew VZ Brower and Randall T Schuh. 2021. *Biological Systematics: Principles and Applications.* Cornell University Press.
[11] Cameron Browne and Frederic Maire. 2010. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games* 2, 1 (2010), 1–16.
[12] Diego Castro and Cláudia Werner. 2021. Rebuilding games at runtime. IEEE, 73–77. https://doi.org/10.1109/ASEW52652.2021.00025
[13] Paul Clements and Linda Northrop. 2002. *Software product lines.* Addison-Wesley Boston.
[14] Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin* 114, 3 (1993), 494.
[15] Norman Cliff. 2014. *Ordinal methods for behavioral data analysis.* Psychology Press.
[16] Jamel Debbiche, Oskar Lignell, Jacob Krüger, and Thorsten Berger. 2019. Migrating Java-based Apo-Games into a composition-based software product line. ACM, 18:1–18:5. https://doi.org/10.1145/3336294.3342361
[17] Mark Gabel and Zhendong Su. 2010. A study of the uniqueness of source code. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Santa Fe, New Mexico, USA) *(FSE '10)*. Association for Computing Machinery, New York, NY, USA, 147–156. https://doi.org/10.1145/1882291.1882315
[18] Roberto Gallotta, Kai Arulkumaran, and L. B. Soros. 2022. Evolving spaceships with a hybrid L-system constrained optimisation evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 711–714. https://doi.org/10.1145/3520304.3528775
[19] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 180, 10 (2010), 2044–2064.
[20] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2019. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.
[21] R. J. Grissom and J. J. Kim. 2005. *"Effect sizes for research: A broad practical approach.* Mahwah, NJ: Earlbaum.
[22] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 1, Article 1 (feb 2013), 22 pages. https://doi.org/10.1145/2422956.2422957
[23] Kent Holsinger and Bruce Weir. 2009. Genetics in geographically structured populations: Defining, estimating and interpreting FST. *Nature reviews. Genetics* 10 (10 2009), 639–50. https://doi.org/10.1038/nrg2611
[24] Robin Hunicke, Marc LeBlanc, Robert Zubek, et al. 2004. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, Vol. 4. San Jose, CA, 1722.
[25] T. Capers Jones. 1984. Reusability in Programming: A Survey of the State of the Art. *IEEE Transactions on Software Engineering* SE-10, 5 (1984), 488–494. https://doi.org/10.1109/TSE.1984.5010271
[26] Bohyun Kim. 2015. Game mechanics, dynamics, and aesthetics. *Library technology reports* 51, 2 (2015), 17–19.
[27] Christoph König, Kamil Rosiak, Loek Cleophas, and Ina Schaefer. 2023. True Variability Shining Through Taxonomy Mining. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A* (Tokyo, Japan) *(SPLC '23)*. Association for Computing Machinery, New York, NY, USA, 182–193. https://doi.org/10.1145/3579027.3608989
[28] Jacob Krüger, Wolfram Fenske, Thomas Thüm, Dirk Aporius, Gunter Saake, and Thomas Leich. 2018. Apo-games: a case study for reverse engineering variability from cloned Java variants, Thorsten Berger, Paulo Borba, Goetz Botterweck, Tomi Männistö, David Benavides, Sarah Nadi, Timo Kehrer, Rick Rabiser, Christoph Elsner, and Mukelabai Mukelabai (Eds.). ACM, 251–256. https://doi.org/10.1145/3233027.3236403
[29] Robert G. Lanergan and Charles A. Grasso. 1984. Software Engineering with Reusable Designs and Code. *IEEE Transactions on Software Engineering* SE-10, 5 (1984), 498–501. https://doi.org/10.1109/TSE.1984.5010273
[30] M. Lenz, H.A. Schmid, and P.F. Wolf. 1987. Software Reuse through Building Blocks. *IEEE Software* 4, 4 (1987), 34–42. https://doi.org/10.1109/MS.1987.231062
[31] Vladimir I Levenshtein. 1965. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmissions* 1, 1 (1965), 8–17.
[32] Crescencio Lima, Wesley K. G. Assunção, Jabier Martinez, Ivan do Carmo Machado, Christina von Flach G. Chavez, and Willian Douglas Ferrari Mendonça. 2018. Towards an Automated Product Line Architecture Recovery: The Apo-Games Case Study. ACM, 33–42. https://doi.org/10.1145/3267183.3267187
[33] Crescencio Lima, Christina Chavez, and Eduardo Santana de Almeida. 2017. Investigating the Recovery of Product Line Architectures: An Approach Proposal, Goetz Botterweck and Claudia Werner (Eds.). Springer International Publishing,

Cham, 201–207.

[34] Crescencio Lima, Ivan do Carmo Machado, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. 2018. Recovering the product line architecture of the apo-games, Thorsten Berger, Paulo Borba, Goetz Botterweck, Tomi Männistö, David Benavides, Sarah Nadi, Timo Kehrer, Rick Rabiser, Christoph Elsner, and Mukelabai Mukelabai (Eds.). ACM, 289–293. https://doi.org/10.1145/3233027.3236398

[35] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. 2021. Deep learning for procedural content generation. *Neural Computing and Applications* 33, 1 (2021), 19–37.

[36] Niko Mäkitalo, Antero Taivalsaari, Arto Kiviluoto, Tommi Mikkonen, and Rafael Capilla. 2020. On opportunistic software reuse. *Computing* 102 (2020), 2385–2408.

[37] Jabier Martinez, Xhevahire Tërnava, and Tewfik Ziadi. 2018. Software Product Line Extraction from Variability-Rich Systems: The Robocode Case Study *(SPLC '18)*. Association for Computing Machinery, New York, NY, USA, 132–142. https://doi.org/10.1145/3233027.3233038

[38] M. D. McIlroy. 1968. Mass-produced software components. *Proc. NATO Conf. on Software Engineering, Garmisch, Germany* (1968).

[39] Michael McShaffry. 2003. *Game Coding Complete*. Paraglyph Publishing.

[40] Alexandre Santos Melotti and Carlos Henrique Valerio de Moraes. 2018. Evolving roguelike dungeons with deluged novelty search local competition. *IEEE Transactions on Games* 11, 2 (2018), 173–182.

[41] Rodrigo André Ferreira Moreira, Wesley KG Assunção, Jabier Martinez, and Eduardo Figueiredo. 2022. Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases. *Empirical Software Engineering* 27, 4 (2022).

[42] P. Naur, B. Randell, and NATO Science Committee. 1969. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October, 1968*. Scientific Affairs Division, NATO. https://books.google.es/books?id=Uc9QAAAAYAAJ

[43] Masatoshi Nei. 1987. *Molecular Evolutionary Genetics*. Columbia University Press.

[44] Cristiano Politowski, Fabio Petrillo, João Eduardo Montandon, Marco Tulio Valente, and Yann-Gaël Guéhéneuc. 2021. Are game engines software frameworks? a three-perspective study. *Journal of Systems and Software* 171 (2021), 110846.

[45] Mike Preuss, Antonios Liapis, and Julian Togelius. 2014. Searching for good and diverse game levels. In *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 1–8.

[46] Naruya Saitou and Masatoshi Nei. 1987. The neighbor-joining method: a new method for Reconstructing Phylogenetic Trees. *Mol Biol Evol* 4, 4 (1987), 406–425.

[47] Victor T. Sarinho, Antônio L. Apolinário, and Eduardo S. Almeida. 2012. A Feature-Based Environment for Digital Games, Marc Herrlich, Rainer Malaka, and Maic Masuch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 518–523.

[48] Jesse Schell. 2008. *The Art of Game Design: A book of lenses*. CRC press.

[49] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.

[50] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186. https://doi.org/10.1109/TCIAIG.2011.2148116

[51] András Vargha and Harold D. Delaney. 2000. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (2000), 101–132. arXiv:http://jeb.sagepub.com/content/25/2/101.full.pdf+html

[52] Grady Weyenberg and Ruriko Yoshida. 2015. Chapter 12 - Reconstructing the Phylogeny: Computational Methods. In *Algebraic and Discrete Mathematical Methods for Modern Biology*, Raina S. Robeva (Ed.). Academic Press, Boston, 293–319. https://doi.org/10.1016/B978-0-12-801213-0.00012-5

[53] G Weyenberg and R Yoshida. 2015. Chapter 12-Reconstructing the Phylogeny: Computational Methods. Algebraic and Discrete Mathematical Methods for Modern Biology. RS Robeva.

[54] Tom Wijman. 2023. *Global Games Market Report*. https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version [Online; accessed 17-November-2023].

[55] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[56] Yuzhong Zhang, Guixuan Zhang, and Xinyuan Huang. 2022. A Survey of Procedural Content Generation for Games. In *2022 International Conference on Culture-Oriented Science and Technology (CoST)*. 186–190. https://doi.org/10.1109/CoST57098.2022.00046

[57] Meng Zhu and Alf Inge Wang. 2020. Model-driven Game Development: A Literature Review. *ACM Comput. Surv.* 52, 6 (2020), 123:1–123:32. https://doi.org/10.1145/3365000