Software Product Families from a Phylogenetics Perspective for Video Game Content Generation

Samuel Navarro[a,b], Jorge Chueca[a,b], Daniel Blasco[a], Carlos Cetina[b], Jaime Font[a]

[a] *Universidad San Jorge, Zaragoza, Spain*
[b] *Universitat Politècnica de València, Valencia, Spain*

Abstract

A Software Product Family comprises similar products within a defined scope that share common characteristics, often due to reuse techniques applied during development. This paper introduces two approaches that apply biological insights to map the landscape of a software product family, identifying potential gaps within its scope, and identifying and extracting features from the family, respectively. Phylogenetics studies the gene similarity among groups of organisms to understand ancestry among species. Leveraging Phylogenetics in software, our approaches offer a structured view of a product family, aiding in the discovery of unexplored areas fitting the scope of the family. Our approaches create 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). Moreover, the phylogenetic tree can be exploited to identify and extract features from the family. Then, those features can be injected into other members of the family. The product family evaluated is a set of industry-scale video game non-playable characters. We assess these approaches through video game simulations and scope metrics to determine how closely the reconstructed and injected products align with the family's scope. The results confirm that the content generated with our phylogenetics-based approaches 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

1. Introduction

Software is mainly created through reuse. Since the term software engineering was coined at the NATO Conference held in Garmisch in 1968 [1], its evolution has been tied to the concept of reuse (also coined during that conference [2]). Either applying an opportunistic approach [3] such as clone-and-own, or applying a systematic approach as the Software Product Lines (SPLs) propose [4].

These practices result in most products being built by reusing existing ones. Several studies [5, 6, 7, 8] have reported reuse percentages across products ranging from 10% to 85% during the late 80s. More recently, the plastic surgeon theory [9] found that 43% of commits to a large repository of Java projects could be reconstituted from existing code. Similarly, Gabel and Sue [10] 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) [11, 12], where creating new content for video games is the bottleneck in an industry that has become the largest entertainment sector, surpassing music and cinema [13] and accounting for one out of two software developers [14].

To address this need for content, some works from the GSE community have applied Procedural Content Generation (PCG) techniques [15, 16] to accelerate the development of new features for their families of products. However, those techniques are primarily based on evolutionary computation [17], 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 [18]. Additionally, interpreting the results is challenging for game developers [19], 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

Email addresses: snavarrod@usj.es (Samuel Navarro), jchueca@usj.es (Jorge Chueca), dblasco@usj.es (Daniel Blasco), cetina@upv.es (Carlos Cetina), jfont@usj.es (Jaime Font)

	SPLC'24 [20]	This work
Input	Product family	Product family
Foundation	Phylogenetic inference	Phylogenetic inference
Leverages	Ancestor → Latent configuration	Ancestor → Latent configuration + Deep ancestor → Feature
Reconstruction	Manual	Manual
Study case	Kromaia	Kromaia

Table 1: Comparison of [20] and this work

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.

This paper is an extension of our previous work [20] in which we presented a phylogenetic-based content generation approach, called Latent Content Generation (LCG). This approach performs a phylogenetic analysis of a family of software products (video game elements from a commercial game), arranges the results into a phylogenetic tree, identifies ancestors, and reconstructs them using their descendants. We evaluated the new content as new family products. We also compared 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. In summary, our main contribution lies in leveraging the analysis of variability among existing products through a phylogentics perspective to generate content for video games.

In addition, this paper leverages deep ancestors to identify features, through a new approach called Feature-Injected Content Generation (FICG). In this novel approach, features identified are injected in other products of the family; then new products are evaluated and compared with our previous approach for content generation and with state-of-the-art approach. In other words, this work goes further in leveraging phylogenetics for software product lines. In fact, this work enables leveraging features of deep ancestors by means of phylogenetic events. A phylogenetic event (also referred to as mutation event) occurs when a branch in the tree diverges into two or more branches [21]. Table 1 compares both studies.

The results of our previous work indicated that the elements generated with our ancestors' approach align more closely with the original family of products' scope than those generated with the state-of-the-art approach. The statistical analysis shows that the differences are significant, and there is a large effect size between LCG approach and the baseline. This paper improves this results with those corresponding to our phylogenetic events' approach, showing lower differences between the content generated with our approach and the family's scope, than the those generated with the state-of-the-art approach. Applying the proposed techniques can create new content through a new perspective on the video game's product line.

Summarizing, this extension further explores the application of phylogenetics to SPLs. Specifically, it introduces a novel method for feature identification through phylogenetic events. The evaluation compares this new approach with both the ancestors' method and a state-of-the-art PCG approach. The results demonstrate that content generated using our phylogenetic events' approach aligns more closely with the original family than both the previous method and the state-of-the-art PCG approach. Statistical analysis confirms that these results are significant.

The paper is structured as follows. Section 2 presents the background for this work. Section 3 presents both Phylogenetic Content Generation approaches. 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 video game Kromaia, and presents phylogenetic inference and its characteristics borrowed from the Biology field and how it is used to create phylogenetic trees in Software Product Families.

2.1. Kromaia's Product Family

In 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) [22] for this task.

Kromaia[1] is a video game released on PC in 2014 [23], and on console in 2015 [24]. 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 [25].

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

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

parametrize behaviors. The original DSL used to specify NPCs in the commercial video game Kromaia is SDML[2] [26].

The main artifact of our approaches is video game content. Specifically, we will use content from Kromaia to evaluate our approaches.

2.1.1. *Kromaia NPCs' Rules*

In Kromaia, all NPCs are defined by a specific set of design rules that define their anatomy, behavior, and overall configuration. These guidelines provide a consistent framework for developers to create new content within the video game. The rules can be summarized as follows:

- **Anatomy:** The anatomical structure maintains the foundational infrastructure of the NPCs in terms of spatial organization. Each NPC is composed of one or more hulls, connected together with links.
- **Visual Appearance :** Each NPC has a set of mesh files associated with its anatomical characteristics, giving every object its visual appearance.
- **Weaponry:** In Kromaia, NPCs can have various types of weapons. The number and type of weapons are crucial decisions developers must make to ensure a fun experience
- **Behavior:** NPCs have different ways of moving in space under specific circumstances. Similar to weaponry, developers must carefully specify these characteristics to ensure NPC creativity.
- **Cosmetic Content:** Some objects are embedded in the anatomy for purely cosmetic purposes, optionally displaying animations.
- **General parameters:** Parameters such as scale or time values, which are present globally in all NPCs, are described by developers to ensure coherence between all components.

These rules are specific to the Kromaia domain and are applied to both our approaches. However, while the rules above refer to the particularities of Kromaia, the idea of rule definition can be extended to other video game domains. In general, these rules should be determined by the development team, based on their understanding of the structure and semantics of the content in their game. The essential requirement is that these rules reflect the internal logic and creative boundaries of the product family, allowing new instances to be generated that remain consistent with existing content.

2.2. *Phylogenetics applied to Video Games*

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 [27], and, in such a field of study, the genetic distance

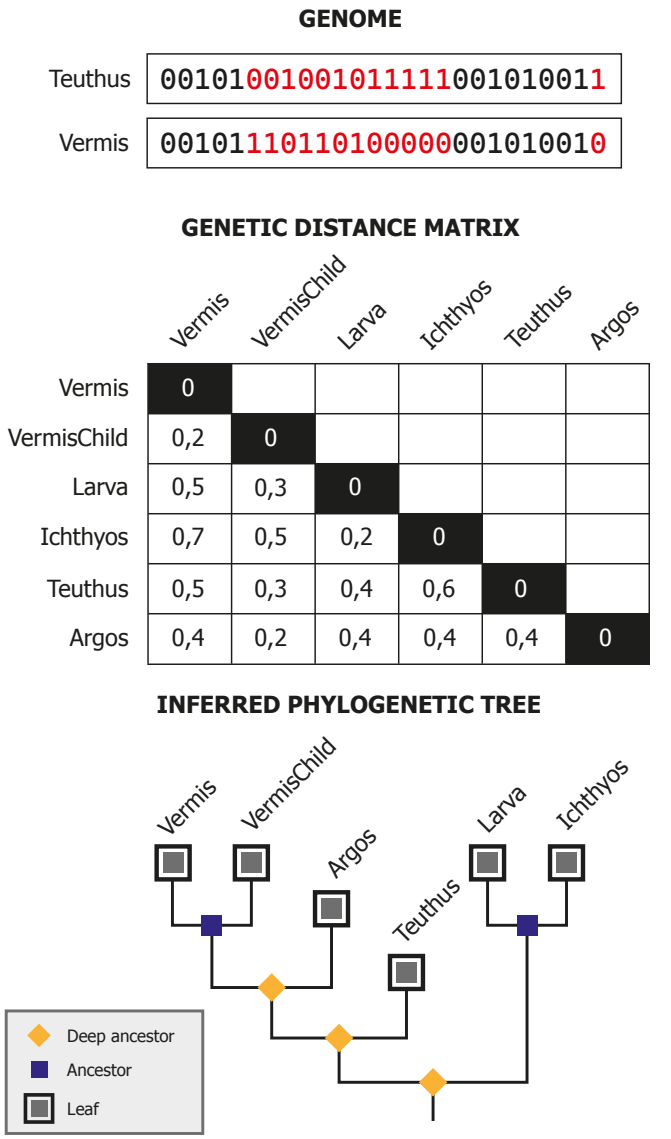

Figure 1: Example of an inferred phylogenetic tree from a distance matrix, with Kromaia's content. At top, a pair of genomes is shown with differences highlighted in red.

is calculated for two specific taxa comparing their genomes. Taxon is a concept that refers to a group of organisms that have a set of genetic characteristics in common. For example, the *Kromaia* family content is a high-level, general taxa that includes all the NPCs in the video game, whereas *Vermis*, is a taxon that refers to a specific NPC only. The comparison of a given set of taxa is commonly done using a data structure known as a Genetic Distance Matrix: a square matrix that represents the genetic distance between each pair of taxa studied [28]. The top part of Figure 1 shows an encoded genome of a sample taxon. The middle part of Figure 1 shows a genetic distance matrix, where the diagonal values are always zero. 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 [29].

[2] Video available at https://svit.usj.es/kromaia-sdml

Inference techniques can be applied to generate a diagram known as Phylogenetic Tree [30]. In this tree, each leaf represents a taxon from the ones studied (i.e., an NPC in the Kromaia's family), while the branches illustrates their relationships in terms of genetic divergence. Moreover, this branches show the ancestry relation between taxa, with internal nodes that hypothesize fossil ancestors. The lower part of Figure 1 presents an example of phylogenetic tree inferred for the *Kromaia* family content. For illustrative purposes, we have highlighted *ancestors* (nodes connecting two leaves) and *deep ancestors* (nodes connected to three or more leaves).

3. Phylogenetic Content Generation

In our previous work [20], the Latent Content Generation approach presented was based on the perspective that phylogenetics can provide to a product family by representing genetic relations among products in a single tree. From this phylogenetic tree, the game developer can identify potential gaps (ancestors) that serve as a seed for creating new content. In this context, a seed refers to a product or configuration that serves as the starting point for content generation. We called this new content as Latent Content.

Moreover, in this work we are presenting the Feature-Injected Content Generation (FICG) approach. The main idea of this approach is that a phylogenetic tree can also be used to identify and extract features, by means of phylogenetic events. Then, the developer can inject these features to create new content. We name this new content as Feature-Injected Content.

Figure 2 presents an overview of our content generation approaches: Latent Content Generation (LCG), and Feature-Injected Content Generation (FICG). In the figure, solid boxes represent operations or calculations, while dotted boxes represent artifacts.

Both approaches begin with the same initial step called Latent Content Identification (LCI). In this step, each product in a family is encoded into a genome-like representation. These encoded products are compared in pairs to create a distance matrix. Based on this matrix, a phylogenetic inference process is applied to produce a phylogenetic tree. The resulting tree is validated by the developers team, to ensure it reflects the structure and relationships within the video game product family, based on their domain knowledge.

At the top of the Figure 2, the second step of the LCG is shown. This second step is called Latent Content Reconstruction (LCR), where the developer identifies ancestors, selects two genetically related products (descendant products) and reconstructs the ancestors, by using a set of domain-specific rules (see Section 2.1.1). The result is the generation of latent content.

At the bottom of the Figure 2, the second and third steps of the FICG are shown. The second step is called Feature Identification & Extraction (FIE), where the phylogenetic events (deep ancestors) are analyzed to identify features. This analysis separates the products into two groups: those that contain the feature, and those that do not. The group with the feature is used to extract it. Then, in the Feature Injection (FI) step, a product without the feature is chosen as a host, and the extracted feature is injected, according to the family rules (see Section 2.1.1). The result is the generation of feature-injected content.

LCG reconstructs latent content, based on the relationships among products, while FICG focuses on identifying and injecting features evolving existing products.

These approaches serve 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, or injecting features extracted from the product family.

3.1. Latent Content Identification

This section presents in more detail the phylogenetic analysis followed in Latent Content Identification process. First, existing content is encoded into genetic information. Second, each taxon is compared with the rest of the taxa, calculating the distance among each pair of taxa, and generating a matrix. Finally, the Phylogenetic Tree is inferred.

3.1.1. Encoding

The first step in the phylogenetic analysis is to encode the content into genetic information. In this work, we apply a string encoding to represent the software models, as others have done for years [26]. 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. Figure 1 shows an example of a simplified string encoding for Argos. In this representation, each character corresponds to an element in the SDML model of the boss from Kromaia. A value of *1* indicates that the element is included in the model, while a *0* denotes its absence.

3.1.2. 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 [31]. The values range 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 [32]. The larger this distance, the smaller the chance that a model is evolved into another model by a series of changes. Each genome is compared one by one, and then the average is computed (i.e., the genome 001 compared with the genome 000 would compute a 0.333, as there is only a gene that differs between both genomes). For example, the genetic distance between Argos and Vermis (see Figure 1) is 0.5, as 13 of the 26 genes differ between them. However, other distances can be used or created in the future.

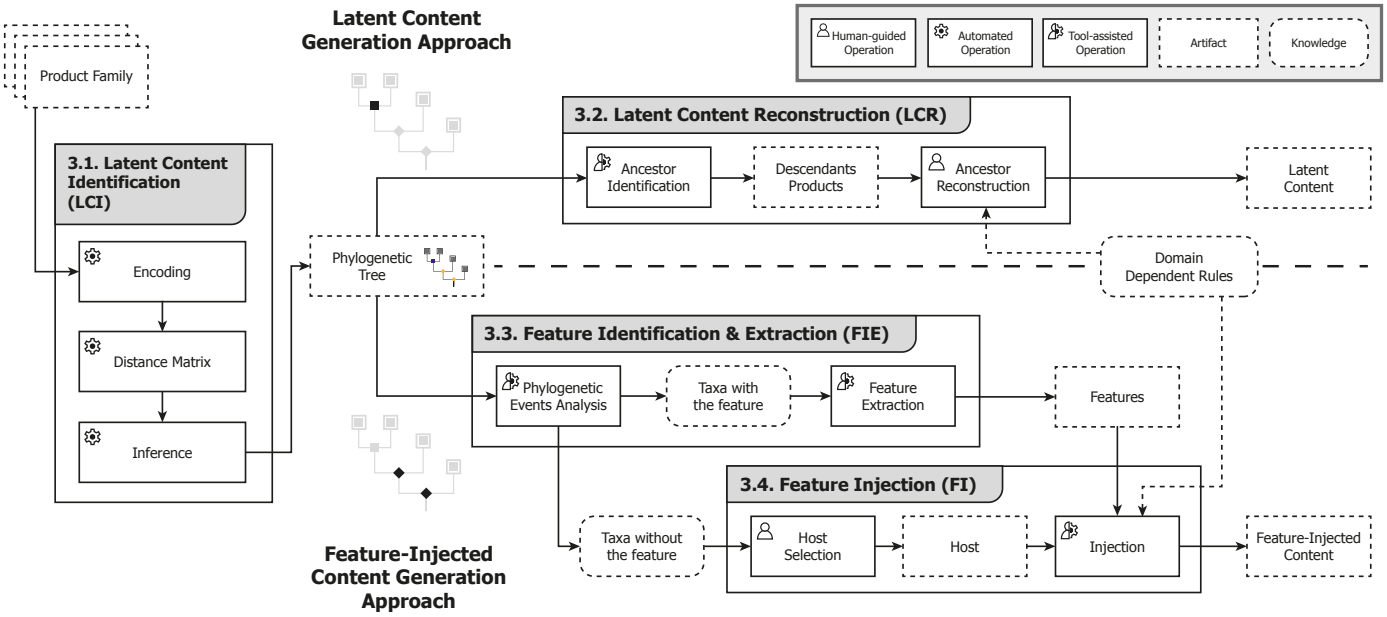

Figure 2: Our Phylogenetic Content Generation Approaches

3.1.3. 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 [33]. 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.

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.

In the genetic distance matrix of Figure 1, we can observe the distances between the bosses of Kromaia. The selection of pairs to merge is done according to the lowest distance. In this case, the pairs Vermis&VermisChild, Ichthyos&Larva, and Argos&VermisChild have a the minimum distance value of the matrix, 0.2. When multiple pairs have the same minimal distance, the method resolves the tie by selecting one of them at random. In the example, the pair selected is Vermis&VermisChild. Now, these two nodes are merged into a new cluster, Vermis+VermisChild, and the matrix is updated accordingly. To compute the distance between this new cluster and the remaining nodes, the method applies an average. For instance, the distance between Vermis+VermisChild and Argos is calculated as the average of the original distances:

$$\frac{\text{Argos\&Vermis} + \text{Argos\&VermisChild}}{2} = \frac{0.4 + 0.2}{2} = 0.3$$

This process continues iteratively until all nodes are merged into a single tree, like the one in Figure 1.

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. Blue squares in Figure 1 represents the gaps corresponding to ancestors. Moreover, those ancestors that are connecting three or more final descendants are represented as yellow diamonds in Figure 1. These ancestors are not present in the original family and serve as hypothetical individuals from which the current individuals could have evolved. 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. In the context of video games, the content is mainly not formalized, specially due to its creative nature. Phylogenetic trees offer a way to find implicit patterns in content, helping developers to formalize the variability and commonality within the family.

3.2. Latent Content Reconstruction

This section explains the Latent Content Reconstruction process. First, the phylogenetic tree from the Latent Content Identification is analyzed, looking for ancestors (Ancestor Identification). Then, the selected ancestors are reconstructed, through the Ancestor Reconstruction operation, using a set of family-rules.

3.2.1. Ancestor Identification

In the phylogenetic tree produced by the Latent Content Identification, leaves represent the products of the family we are analyzing. The inner nodes indicate phylogenetic events (divergence points). Among these, the nodes that are directly connected to two terminal nodes (i.e., two family products) are re-

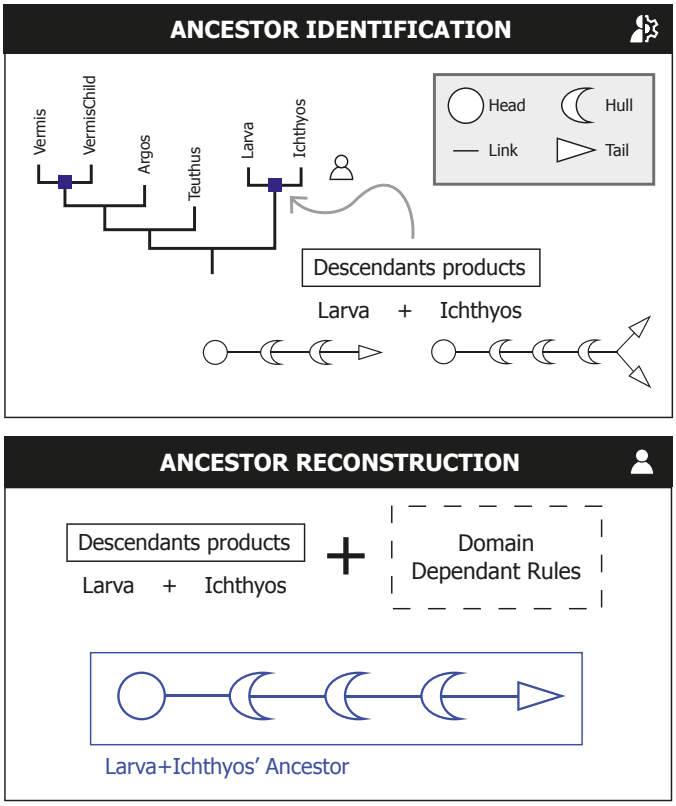

Figure 3: Latent Content Reconstruction steps.

ferred to as ancestors (see blue square nodes in Figure 3). Each of these ancestors represents a hypothetical product, a proposed common origin from which the two existing products are derived. This type of node is known as the Most Recent Common Ancestor (MRCA) [34].

The possible ancestors of a products family is programmatically extracted from the phylogenetic tree. However, the developers must determine which of the identified ancestors are suitable candidates for reconstruction, based on domain-specific criteria such as design intent, semantic coherence, or implementation order. In the context of Kromaia, the developers must consider aspects like the age and antecedence in relation to elements like files or weapon/projectile types, based on their implementation dates or on which type can reasonably be considered the precursor of the other two. After verifying the eligibility of the identified ancestors, the Ancestor Reconstruction process can begin. This process can be performed multiple times, one for each feature to extract.

Figure 3 shows a simple example of the Ancestor Identification process. First, an algorithm identifies all the possible ancestors of the phylogenetic tree. Then, the developers analyze the ancestors identified, and decides which are able to be reconstructed, keeping consistency with the rest of the game. In that example, only the ancestor of Larva and Ichthyos has been selected for reconstruction.

3.2.2. Ancestor Reconstruction

The reconstruction of an ancestor is done manually by the video game developers, based on the set of rules used to define the content of the family. In the case of Kromaia, these rules specify aspects such as the anatomy, visual appearance, weaponry, behavior, and other general parameters of the game's bosses (see Section 2.1.1).

For each selected ancestor, developers analyze the SDML models of the descendant NPCs. The reconstruction process involves integrating features from both descendants to create a new one that represents the ancestor. Ensuring the internal coherence, functional plausibility, and consistency with the overall game design is essential. Therefore, video game developers also take into account additional factors, such as the weaponry/projectile types, for example. A reconstructed ancestor from Larva and Ichthyos products can be seen in Figure 3.

3.3. Feature Identification & Extraction

The Feature Identification & Extraction process is composed by two steps. First, a phylogenetic events analysis is performed to find features, and determine two sets of products (the ones with the feature, and the ones without it). Second, each feature identified is extracted. This section presents in more detail both steps: Phylogenetic Events Analysis, and Feature Extraction.

3.3.1. Phylogenetic Events Analysis

While the Latent Content Reconstruction process uses ancestors, the Feature Identification & Extraction process uses deep ancestors, the MRCA of three or more products. Yellow diamonds in Figure 1 are deep ancestors. Each deep ancestor represents a phylogenetic event that can be encoded as a genome sequence. The Phylogenetic Events Analysis consists of two steps.

First, for each deep ancestor, a genome is programmatically generated by comparing the genomes of its descendants, and merging their common genes (common factor). Second, the developers analyze the genomes generated to identify features. The selection of deep ancestors to analyze is not arbitrary; developers leverage their domain knowledge about the content to guide this process, focusing on the most promising candidates rather than exhaustively exploring all deep ancestors.

When the feature is identified, the taxa of the tree is automatically splitted into two group: taxa with the feature, and taxa without it. In the selection of deep ancestors, the root ancestor of the phylogenetic tree is excluded, as it does not allow for a meaningful division into two subsets of taxa. Instead, it would result into a single subset with all taxa that contain the feature identified.

Figure 4 shows an example of this process. The deep ancestors of the tree are processed by the algorithm, and the genomes are produced (see the case of Vermis ∩ VermisChild ∩ Argos). The developers, using their knowledge, select the deep ancestor of Vermis ∩ VermisChild ∩ Argos, as all its descendants share a common feature (Double-arm feature), that is not present in the rest of the game. Finally, the taxa is divided into taxa with Double-arm (Vermis, VermisChild and Argos), and taxa without it (Teuthus, Larva and Ichthyos).

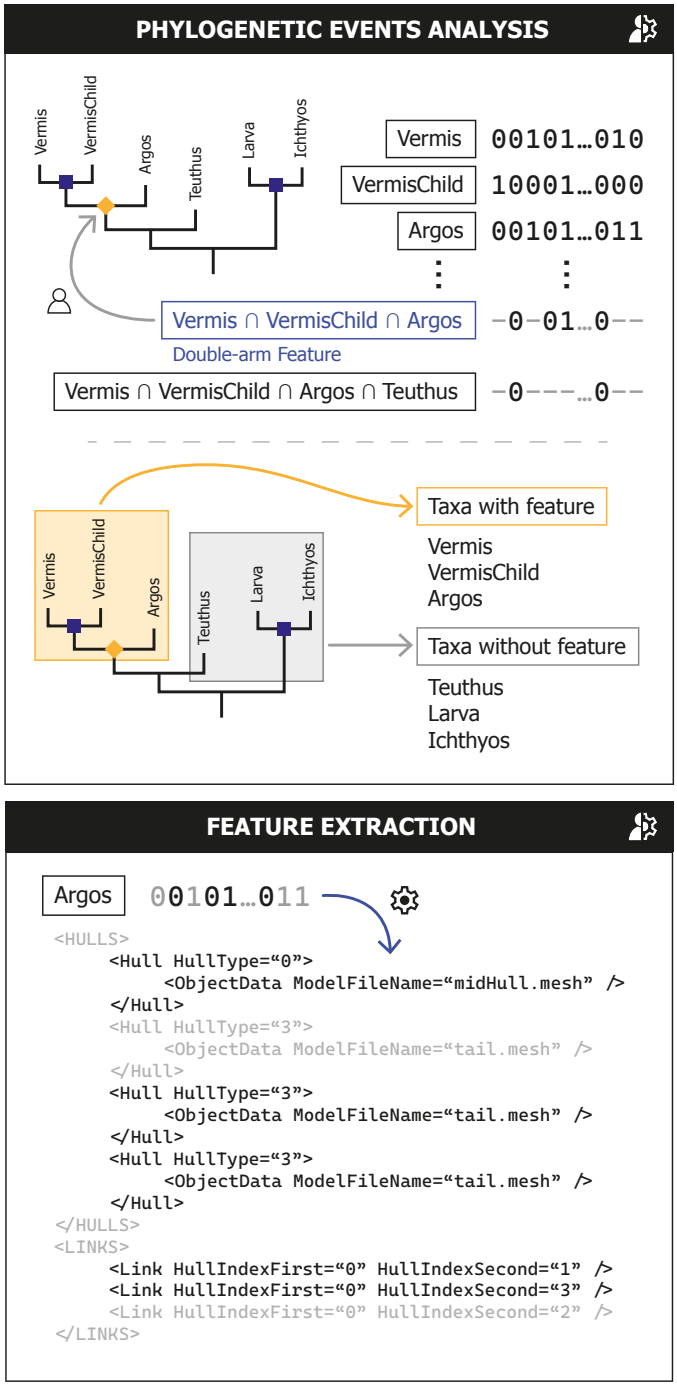

Figure 4: Identification and Extraction steps for *Double-arm* feature.

3.3.2. *Feature Extraction*

In the Feature Extraction process, the genome of a product with the feature is decoded into a set of elements and rules that characterize the feature, following a reverse process than in the encoding step. These characteristics and rules define the anatomy, behavior, and other visual and functional aspects of the products where the feature will be injected later.

To perform this process, the developers must select one of the products with the feature, ensuring that it is a good representative of all products sharing that feature. This selection enhances the accuracy of the results during the decoding phase.

In Figure 4, an example of this process is presented. In that case, for the feature Double-arm, the developers have selected the product Argos, which represents this feature. Then, the genome of Argos is decoded, resulting in a set of lines in SDML model format, that represent the extracted feature.

3.4. *Feature Injection*

From the phylogenetic tree we have extracted features, and identified those products in the family that include each feature, and those that do not. In this step, the developers proceed with the injection of the features in some of the individuals without it. This process is inspired by transplantation techniques [35, 36]. In Biology, transplantation refers to a procedure in which organs of a host are replaced by those of a donor [37]. In software terms, researchers understand a fragment of a software element as an organ of a donor [35]. In the context of Kromaia, this process involves transplanting a feature (the organ) from a product (the donor) into another product (the host) that does not include it. Specifically, we used the transplantation process proposed for SDML model fragments (the models used in our case study) [36]. This section presents the operations performed to execute the Feature Injection process: Host Selection, and Injection. Figure 5 shows the steps of Feature Injection for an hypothetical *Double-arm* feature.

3.4.1. *Host Selection*

The first step in the Feature Injection process involves identifying suitable hosts (i.e., taxa that currently lack the feature but are appropriate candidates to receive it). For each feature to be injected, the developer selects a host from products without the feature, considering their compatibility with the existing game content. For example, in Figure 5, the developers consider the bottom-right candidate to be the most suitable, as it is expected to produce a new enemy that remains balanced with the rest of the enemies in Kromaia.

3.4.2. *Injection*

Once suitable hosts have been identified, the Injection of the feature in the chosen product is performed.

In this process, the transplantation algorithm detects boundaries in donor product (determining the limits of the feature) and in host product (determining possible feature destinations). Then, a mapping between feature and the host boundaries is also realized by the algorithm. Finally, the developers transplant the feature, according to the reconstruction rules of the product family (see Section 2.1.1).

To ensure a seamless integration, the feature must be adjusted to fit the host, maintaining coherence in terms of visual appearance, functionality, and general parameters of the video game content. The developers select the most suitable connections, deciding the boundaries where the feature is injected. The goal is to produce a consistent and enjoyable member of the product family that fits within the game experience.

Figure 5 shows the boundaries identified in both the donor (d) and the host (h). The feature *Double-arm* is delimited by

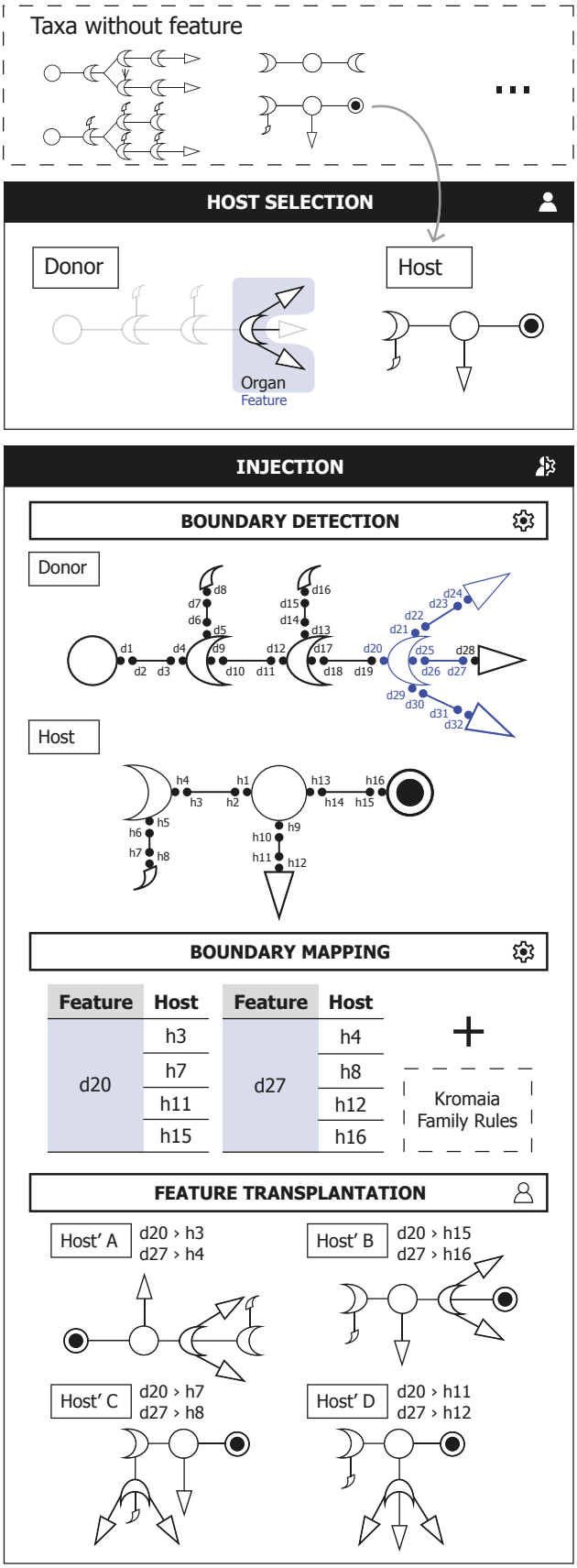

Figure 5: Injection process steps for *Double-arm* feature injection.

the boundaries *d20* and *d27*. These boundaries are then mapped to potential connection points in the host. Specifically, *d20* can be connected to *h3*, *h7*, *h11* or *h15*, while *d27* can be linked to *h4*, *h8*, *h12* or *h16*. The developers apply the Kromaia Family Rules to determine the connection points that best fit the host product, and then performs the injection of the feature. The figure shows four possible outcomes of the selected host with the injected feature.

4. Evaluation

To evaluate our approaches, 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?
- **RQ4** - Does our Feature-Injected Content Generation approach produce results that fit the scope of the family better than the state-of-the-art approach for PCG?
- **RQ5** - If so, how big are the differences between both approaches?

In this work, the generated content is evaluated using video game simulations, an accepted practice [38] 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).

This section is structured as follows. Section 4.1 presents the approaches evaluated. Section 4.2 presents the Experimental Setup for the evaluation. Section 4.3 presents how game simulations work for content evaluation. Section 4.4 presents the metrics used to compare with. Section 4.5 presents the details of the implementation. Section 4.6 presents the results. And Section 4.7 presents the statistical analysis.

4.1. Content Generation Approaches

To answer the research questions, we compare our LCG and FICG approaches 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. [39] 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 [39]. The approach from Gallota et al. is

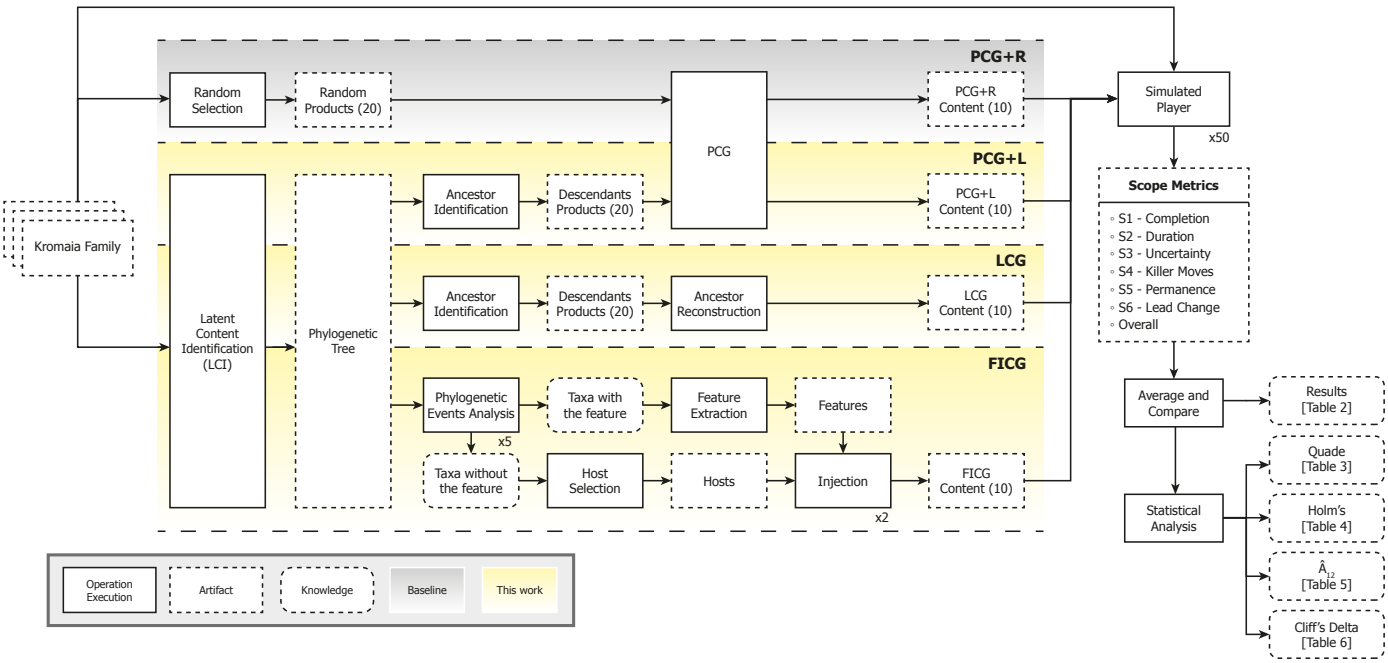

Figure 6: Experimental setup for the evaluation and comparison of four approaches: PCG+R, PCG+L, LCG, and FICG. The flow starts in the Kromaia Family. Each approach is represented in a line. The new content is evaluated using a simulated player. Results of the simulation and statistical analysis can be found in Tables 2, 3, 4, 5, and 6

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 LCI 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 four sets of new content generated (see the middle part of Figure 6):

- **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.
- **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.
- **FICG**: Content created by our Feature-Injected Content Generation approach. It is the result of applying the Feature Identification & Extraction operation to the phylogenetic tree, followed by the Feature Injection process.

4.2. Experimental Setup

Figure 6 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. The family is fed as input to the four content generation approaches: the state-of-the-art approach for PCG used as a baseline (PCG+R), a version of the baseline enhanced using our Phylogenetic operation (PCG+L), our Latent Content Generation approach (LCG), and our Feature-Injected Content Generation approach (FICG).

In the PCG+R approach, 20 enemies are selected randomly. These enemies feed the state-of-the-art PCG operation, generating 10 new enemies.

In the PCG+L approach we enhanced the PCG algorithm. Instead of selecting randomly 20 enemies, 10 pairs of parents (20 enemies, depicted as *Descendant Products*) are selected using the phylogenetic tree from the LCI process. Then, selected enemies fed the state-of-the-art PCG approach, generating 10 new enemies.

In our LCG approach, the phylogenetic tree is used to identify 10 suitable ancestors, each with two direct descendants. These descendant pairs are used as input for the reconstruction operation, resulting in the creation of 10 new enemies.

In our FICG approach, the phylogenetic tree is used to analyze deep ancestors for the identification and extraction of features. From the phylogenetic tree, we have extracted 5 features by analyzing the phylogenetic events, and using taxa with and without each feature. For every extracted feature, 2 hosts with-

out it were selected for injection, resulting in 10 new enemies generated in total.

Then, the content generated by the four 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 (*S1-Completion*, *S2-Duration*, *S3-Uncertainty*, *S4-Killer Moves*, *S5-Permanence*, and *S6-Lead Change*) [26]. In addition, the original content used in the evaluation is also assessed using those metrics (see top arrow in Figure 6). This allows us to compare the generated content with the original content, using a common set of metrics, in order to determine whether the new content remains consistent with the design and gameplay scope of the existing material.

Finally, the data is processed to present the results of the evaluation (see right-bottom part of Figure 6). 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 Post Hoc) [40] and effect size ($\hat{A}_{12}$ and Cliff's Delta) [41, 42, 43].

4.3. Game Simulations

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, which is within the suggested range in the literature [44].

4.4. 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.

The content of the video game must be properly aligned with the intended player experience, empowering game designers to effectively convey that experience through gameplay[45]; mechanics, dynamics, and aesthetics of a video game are interrelated [46], 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) [26, 38]. Formula for each metric is defined in [26].

Our evaluation first measures the six criteria for the family of content from Kromaia (clamped for standardization purposes [26]). Then, we measure the six criteria for the generated content. Finally, we calculate the distance between the generated content and the original content from the family. A lower distance indicates a better alignment with the scope. Since these indicators are used in the context of the "scope" concept, we adopt a naming convention for each metric, using the prefix **S** followed by a unique identifier for the metric.

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). A timeout is defined by the game designers. Depending on the type of game, duels can be designed to last longer or shorter, so timeout can be adapted. When a simulation times-out, duel is considered as a tie. 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:

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

A conclusion is counted when the duration of the duel (T_d) is strictly less than the timeout set by the developer:

$$Conclusions = \#(d \in Duels \mid T_d < timeout) \tag{2}$$

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 duels 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 ($T_{Optimal}$):

$$\text{S2-Duration} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels}\right) \tag{3}$$

S3-Uncertainty: If a duel outcome can be foreseen in advance, the player might lose interest when realizing that they will lose or win 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).

$$\text{S3-Uncertainty} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min(P_d, B_d)}{T_d}}{Duels}\right) \quad (4)$$

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 can determine the duel's outcome. *S4-Killer Moves* measures the ratio between killer moves (K) and remarkable highlights (H).

$$\text{S4-Killer Moves} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels}\right) \quad (5)$$

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:

$$\text{S5-Permanence} = clamp_{[0,1]}\left(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d + K_d}}{Duels}\right) \quad (6)$$

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:

$$\text{S6-Lead Change} = clamp_{[0,1]}\left(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d + K_d}}{Duels}\right) \quad (7)$$

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:

$$\text{Overall} = \frac{\sum_{i=1}^{N} S_i}{N} \quad (8)$$

4.5. Implementation Details

The evaluation has been performed using a commercial HP laptop with an Intel i7-10750H processor, 16Gb of RAM, and Windows 11 64 bits as the host operating system. The presented approach for Latent Content Generation has been implemented using the .NET 6 runtime environment and C# 10 as the programming language. 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 Ancestor Identification step, guided by the developer, was completed in approximately 5 minutes. The Ancestor Reconstruction, which is performed manually by the developers, required around 2 hours. The Phylogenetic Events Analysis took about 15 minutes and was also guided by the developers. Feature Extraction was completed in less than 1 minute: once the developer selects the representative product, the decoder extracts the feature automatically in a few seconds. Host Selection is a manual step and its duration can vary significantly depending on the complexity of the products involved and the feature to be injected. Injection step is typically completed in a few minutes (the automated operations run in seconds, and the feature transplantation can vary depending on the complexity of the mappings). Finally, the evaluation performed via Simulated Player takes around 30 seconds to complete.

The metrics used in the evaluation 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%. These parameters are used to evaluate both the original and generated content during player simulations.

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[3].

4.6. 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 7. This tree has a structured view of the product's family and the genetic relations between products. It serves as the starting point for the reconstruction of latent content, as well as for the feature identification process.

For the evaluation of LCG approach, we exploited the phylogenetic tree generated with the LCI operation (Figure 7). The

[3] https://svit.usj.es/rp/Navarro_JSS_2025

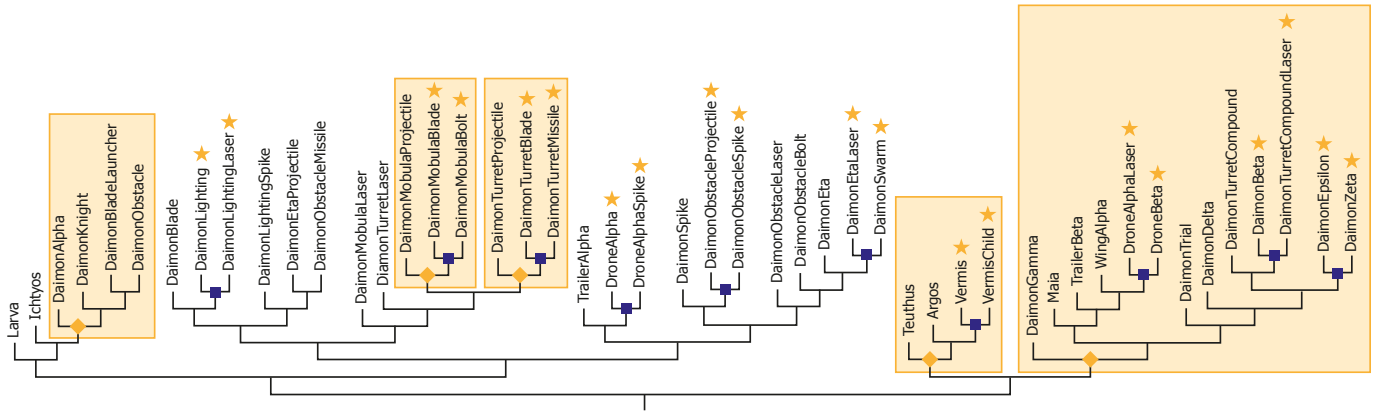

Figure 7: Phylogenetic Tree of the family of enemies from Kromaia. Each enemy used as input is tagged with a star; ancestors used in the LCG process are identified with blue squares; deep ancestors used for Feature Identification & Extraction are depicted as yellow diamonds; taxa with features identified are depicted as yellow areas (each area for each feature).

	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
FICG	0.00003 ± 0	0.07021 ± 0.05946	0.01305 ± 0.00055	0.0445 ± 0.03600	0.00609 ± 0.00202	0.02285 ± 0.03551	0.01034 ± 0.01002
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 2: 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.

phylogenetic tree provides descendants from the tree's leaves (blue rectangles) and its ancestor to be reconstructed (line-joins). Each ancestor used in this evaluation is represented as a blue square. 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.

In the case of FICG approach, deep ancestors of the phylogenetic tree are exploited to identify and extract features. Deep ancestors used in this evaluation are represented as yellow diamonds, and taxa with each feature identified and extracted are highlighted also in yellow. As in the case of LCG, the selection of deep ancestry was performed by an expert video game developer of Kromaia, with a corroboration performed by another Kromaia developer.

For example, the group consisting of DaimonBladeLauncher, DaimonObstacle, DaimonKnight and DaimonAlpha (first group from left to right), corresponds to the *Hair* feature, as defined by the professionals. Taxa with this feature have embedded in their anatomy, strands of hair, acting purely as cosmetic content.

Table 2 presents the results obtained by the simulations executed for each NPC created, and for each product in the Kromaia family. The first row of values shows the average value of the simulations realized with the original content of Kromaia. The "Scope" values are used as reference for the rest of the approaches. The next rows present the differences between the results obtained for each approach, and the reference value (i.e., scope differences). Therefore, lower values are considered more favorable. The results show that our approaches (LCG and FICG) allows the creation of new content more similar to the NPCs in video games, than those created with the baseline's approaches. The differences between our LCG approach and the reference scope metrics from the family (see the first row of Table 2) are smaller than those created by the baselines (PCG+R and PCG+L). Moreover, the differences between the reference values and our FICG approach are even smaller than the differences between the reference values and our LCG approach.

We can see that the *S1-Completion* metric is identical to the scope in the four approaches measured. *S1-Completion* is the minimum requirement an enemy should meet, for it is expected that the battles, at least, can be completed. If these metrics differ significantly with the scope, the content generated is considered invalid for the game. Moreover, these high values come from the characteristics of the battles, where the effectiveness of the weapons, and the difficulty of an avarage player to avoid combat, maximize the duel conclusions within duration limits [47]. 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 and FICG start 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 and FICG are 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, and more than three times more in scope in favor of FICG.

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

4.7. Statistical Analysis

To address RQ2 and RQ5, 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 [44].

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.

4.7.1. 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 represents the probability of validating the null hypothesis (H_0). In this field, a p-value under 0.05 leads to the rejection of H_0 in favor of H_1, indicating that exists an statistical significance between, at least, two of the content generation approaches evaluated.

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 [40].

Table 3 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 can only conclude if some approaches produce different outputs, but does not state which ones. We need to compare the results of each pair of approaches separately. Holm's Post Hoc does this, another statistical test commonly used in conjunction with Quade [40]. The test is run for each metric separately, omitting the *S1-Completion* metric as Quade already showed that differences were not significant.

Table 4 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-Duration*, *S3-Uncertainty*, *S4-Killer Moves*, *S5-Permanence*, *S6-Lead Change*, and *Overall*). In the same way, the differences between FICG and the PCG approaches are also significant for the six metrics. According to Holm's Post Hoc, the differences between both PCG approaches, with latent input or with random input, are not significant except for *S3-Uncertainty*. In addition, in the Table 4 we can see that the differences between LCG and FICG approaches are significant only for the metrics *S5-Permanence*, *S6-Lead Change*, and *Overall*.

4.7.2. 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 [48]. 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 $\hat{A}_{12}$ [41] and Cliff's Delta [42, 43].

The $\hat{A}_{12}$ statistic estimates the probability that a randomly selected value from one distribution will be greater than a randomly selected value from another [41]. Table 5 shows the $\hat{A}_{12}$ statistic for each scope metric (columns) and pair of generation approaches (rows). For example, the second row, the third column in Table 5 shows the $\hat{A}_{12}$ value for FICG vs PCG+L for the *S2-Duration* metric, 25.00%. It indicates the probability that an observation (a randomly selected duel simulated) from the first group (FICG) 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 25.00% of the runs, the differences in scope between the FICG results and the reference values in terms of the *S2-Duration* 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 75.00% of the runs, the differences in the scope of FICG are smaller than the differences of PCG+L for the *S2-Duration* metric.

Cliff's Delta is another non-parametric effect size measure that quantifies the amount of difference between two distributions, by computing the probability that one randomly selected observation is larger than another, minus the reverse probability [42, 43]. Table 6 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). Since no significant differences were found for the *S1-Completion* metric (as the statistical significance test was inconclusive; see Section 4.7.1), the effect size analysis for this metric was not performed.

	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
Quade	-	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$7.0x10^{-8}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$

Table 3: Quade test p-values. Values below 0.05 are highlighted

		S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
Holm's	FICG vs LCG	-	0.21	0.468	0.45	$7.8x10^{-7}$	$\ll 2x10^{-16}$	0.0063
	FICG vs PCG+L	-	$\ll 2x10^{-16}$	$6.1x10^{-7}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	FICG vs PCG+R	-	$\ll 2x10^{-16}$	$\ll 2.1x10^{-14}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	LCG vs PCG+L	-	$\ll 2x10^{-16}$	$\ll 1.8x10^{-5}$	$\ll 2x10^{-16}$	0.00198	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	LCG vs PCG+R	-	$\ll 2x10^{-16}$	$3.9x10^{-12}$	$\ll 2x10^{-16}$	0.00035	$\ll 2x10^{-16}$	$\ll 2x10^{-16}$
	PCG+L vs PCG+R	-	0.21	0.015	0.45	0.57	0.87	0.118

Table 4: Holm's Post Hoc p-values. Values below 0.05 are highlighted

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 four 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). Moreover, for FICG, we can conclude that the content created fits better the family's scope than PCG+L and PCG+R, with a large effect size in both cases.

5. Answer to RQs and Discussion

This section will answer the the research questions described in Section 4, analyzing the results obtained in our evaluation.

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?

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 2).

RQ2 - If so, how big are the differences between both approaches?

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 5). This percentage corresponds to the complement of 24.22%, which represents the cases where the baseline performed better. This can be interpreted as a medium/large difference according to the Cliff's Delta effect size (see Table 6).

RQ3 - Does the Latent Content Identification operation produce better seeds than random for the state-of-the-art approach for PCG?

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.

RQ4 - Does our Feature-Injected Content Generation approach produce results that fit the scope of the family better than the state-of-the-art approach for PCG?

As an answer to RQ4, our FICG approach produces better results than the state-of-the-art approach. The content generated with FICG fits the scope of the family better than the state-of-the-art approach for PCG, in terms of the metrics studied.

RQ5 - If so, how big are the differences between both approaches?

As an answer to RQ5, the results obtained by our FICG approach are bettwer than those obtained by the state-of-the-art approach, generating content with smaller scope differences 78.55% of the times for the overall metric, according to the $\hat{A}_{12}$ effect size (see Table 5). Moreover, according to the Cliff's Delta effect size, this can be interpreted as a large difference (see Table 6).

Our results demonstrate that content created with LCG and FICG are more similar to the already present content than the state-of-the-art techniques for PCG. 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. However, while the LCG approach provides a coarse level of detail by reconstructing ancestors, the FICG method offers a finer granularity. This allows video game's developers to identify and extract specific features from the family, and then inject them into other entities.

Additionally, the structured view that the tree provides can help developers understand the product family's relations and scope thus easing software reuse and variability management. 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

$\hat{A}_{12}$	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
FICG vs LCG	-	51.41%	49.23%	44.26%	40.57%	23.42%	48.90%
FICG vs PCG+L	-	25.00%	44.86%	23.44%	36.42%	14.55%	24.87%
FICG vs PCG+R	-	20.03%	42.00%	25.70%	37.68%	17.60%	21.45%
LCG vs PCG+L	-	23.40%	45.58%	26.70%	44.91%	25.76%	26.55%
LCG vs PCG+R	-	18.31%	42.74%	31.10%	45.47%	27.43%	24.12%
PCG+L vs PCG+R	-	44.62%	47.26%	51.98%	50.03%	51.09%	47.55%

Table 5: $\hat{A}_{12}$ statistic. Each value indicates the probability that the second approach outperforms the first in pairwise comparisons. A 100% value means the second approach always performs better, while lower values favor the first approach. Values below 30% are highlighted, indicating a notable advantage for the first approach.

Cliff's Delta	S1 - Completion	S2 - Duration	S3 - Uncertainty	S4 - Killer Moves	S5 - Permanence	S6 - Lead Change	Overall
FICG vs LCG	-	0.028248 (negligible)	-0.01536 (negligible)	-0.114888 (negligible)	-0.188512 (small)	-0.531624 (large)	-0.021912 (negligible)
FICG vs PCG+L	-	-0.499872 (large)	-0.102792 (negligible)	-0.531224 (large)	-0.271512 (small)	-0.708936 (large)	-0.502648 (large)
FICG vs PCG+R	-	-0.599488 (large)	-0.159896 (small)	-0.486016 (large)	-0.246544 (small)	-0.64804 (large)	-0.570984 (large)
LCG vs PCG+L	-	-0.532072 (large)	-0.088312 (negligible)	-0.465984 (medium)	-0.10184 (negligible)	-0.484864 (large)	-0.468916 (medium)
LCG vs PCG+R	-	-0.633744 (large)	-0.14512 (negligible)	-0.378092 (medium)	-0.090664 (negligible)	-0.451472 (medium)	-0.517696 (large)
PCG+L vs PCG+R	-	-0.10768 (negligible)	-0.054824 (negligible)	0.0396 (negligible)	0.000544 (negligible)	0.021892 (negligible)	-0.04898 (negligible)

Table 6: Cliff's Delta statistic. These values are interpreted, based on their absolute magnitude, as follows: negligible (< 0.11), small ($0.11- < 0.28$), medium ($0.28- < 0.43$), and large (≥ 0.43) [49]. Medium and large effect sizes are highlighted.

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. [50] 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. Similarly, would the common ancestor between the frog, the turtle, and the cobra be helpful in finding a shared feature that can be included in other individuals? Our results say yes. For example, with our FICG approach, we can identify the feature of thriving in both aquatic and terrestrial environments. FICG empowers developers by allowing them to extract such features from the family and inject them into other individuals, creating more versatile and dynamic characters within the game, and taking the first step towards formalization of the variability existing among the family of software products.

6. Threats To Validity

To address the limitations of our evaluation, we adopt the validity threat classification framework from Wohlin et al. [51], 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 [38, 26].

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 an 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 [44] including statistical significance (Quade test and Holm's Post Hoc) and effect size (Cliff's Delta and $\hat{A}_{12}$) statistics.

Moreover, while our evaluation compares fully automated and phylogenetically guided approaches (which involve human intervention), we did not include a baseline based on a purely human-driven content creation process. Although the developers involved were not biased toward any specific outcome,

and simulation-based metrics helped provide objective evaluations, further studies should explore whether similar quality could be achieved without phylogenetic guidance. Including such a baseline in future work could help isolate the specific contribution of the phylogenetic analysis to the quality and efficiency of the process.

Finally, one limitation of these approaches is that they need an existing set of products to create the phylogenetic tree. This is not the case for other PCG approaches. Our approaches 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 [52], an extension of their previous engine called FEnDiGa [53]. 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 [54] 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. [55, 56] introduce two studies concerning the recovery of Product Line Architecture (PLA). They implemented their previously proposed guidelines [57] to establish the PLA for the Apo-Games project [58]. Moreira et al. [59] 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. [60] 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. [61] 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* [62]. 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. [63] 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. [64] characterized quality diversity as a pivotal search strategy within search-based PCG.

The study conducted by Melotti et al. [65] 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 [26, 47]. 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

An study by König et al. [50] 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.

Blasco et al. established the foundational principles of applying phylogenetics to software engineering through a Phylogenetics-aware Search-Based Software Engineering (SBSE) approach in [66]. This work lays the groundwork for the method by defining a phylogenetic operation for evolutionary algorithms. However, it does not focus on feature location within video games.

Moreover, to the best of our knowledge, except our previous paper [20] that this works extends, there is only another study that applies phylogenetics to software. Navarro et al. [67] presented Phylogenix: a tool for Unity that allows video games' developers to phylogenetically analyze the *prefabs* of an existing video game, creating a phylogenetic tree. However, this tool does not generate new content for video games, neither analyze the phylogenetic tree for feature identification or extraction.

8. Conclusion

Our approaches prove themselves to be valid for content creation in video games. The evaluation has demonstrated the capabilities of LCG to identify gaps within a product family, hidden content waiting to be discovered. In addition, the evaluation has shown the possibilities that FICG approach has, allowing developers to identify and extract features from the existing content of the family. The content created via LCG and FICG are more similar to the current scope of an industry-scale video game, than the content created with state-of-the-art PCG aproaches.

Phylogenetics can help in managing 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 approaches 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.

9. Acknowledgements

This work has been partially supported under the projects PHYLOVAR (PID2024-162114OB-C21, PID2024-162114OA-C22) and VARNETICA (CNS2023-145422), through funds from MICIU/ AEI / 10.13039/501100011033 / FEDER, UE. This work was also financed by Gobierno de Aragón (Spain) (Research Group T61_23R), and supported 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] P. Naur, B. Randell, N. S. Committee, Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October, 1968, Scientific Affairs Division, NATO, 1969.
URL https://books.google.es/books?id=Uc9QAAAAYAAJ

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

[3] N. Mäkitalo, A. Taivalsaari, A. Kiviluoto, T. Mikkonen, R. Capilla, On opportunistic software reuse, Computing 102 (2020) 2385–2408.

[4] P. Clements, L. Northrop, Software product lines, Addison-Wesley Boston, 2002.

[5] T. C. Jones, Reusability in programming: A survey of the state of the art, IEEE Transactions on Software Engineering SE-10 (5) (1984) 488–494. doi:10.1109/TSE.1984.5010271.

[6] T. Biggerstaff, C. Richter, Reusability framework, assessment, and directions, IEEE Software 4 (2) (1987) 41–49. doi:10.1109/MS.1987.230095.

[7] R. G. Lanergan, C. A. Grasso, Software engineering with reusable designs and code, IEEE Transactions on Software Engineering SE-10 (5) (1984) 498–501. doi:10.1109/TSE.1984.5010273.

[8] M. Lenz, H. Schmid, P. Wolf, Software reuse through building blocks, IEEE Software 4 (4) (1987) 34–42. doi:10.1109/MS.1987.231062.

[9] E. T. Barr, Y. Brun, P. Devanbu, M. Harman, F. Sarro, The plastic surgery hypothesis, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 306–317. doi:10.1145/2635868.2635898.
URL https://doi.org/10.1145/2635868.2635898

[10] M. Gabel, Z. Su, A study of the uniqueness of source code, in: Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 147–156. doi:10.1145/1882291.1882315.
URL https://doi.org/10.1145/1882291.1882315

[11] A. Ampatzoglou, I. Stamelos, Software engineering research for computer games: A systematic review, Information and Software Technology 52 (9) (2010) 888 – 901. doi:https://doi.org/10.1016/j.infsof.2010.05.004.

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

[13] C. Politowski, F. Petrillo, J. E. Montandon, M. T. Valente, Y.-G. Guéhéneuc, Are game engines software frameworks? a three-perspective study, Journal of Systems and Software 171 (2021) 110846.

[14] T. Wijman, Global games market report, [Online; accessed 17-November-2023] (2023).
URL https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version

[15] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: A survey, ACM Trans. Multimedia Comput. Commun. Appl. 9 (1) (feb 2013). doi:10.1145/2422956.2422957.
URL https://doi.org/10.1145/2422956.2422957

[16] Y. Zhang, G. Zhang, X. Huang, A survey of procedural content generation for games, in: 2022 International Conference on Culture-Oriented Science and Technology (CoST), 2022, pp. 186–190. doi:10.1109/CoST57098.2022.00046.

[17] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (3) (2011) 172–186. doi:10.1109/TCIAIG.2011.2148116.

[18] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via ma-

chine learning (pcgml), IEEE Transactions on Games 10 (3) (2018) 257–270.
[19] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, J. Togelius, Deep learning for procedural content generation, Neural Computing and Applications 33 (1) (2021) 19–37.
[20] J. Chueca, D. Blasco, C. Cetina, J. Font, Leveraging phylogenetics in software product families: The case of latent content generation in video games, in: Proceedings of the 28th ACM International Systems and Software Product Line Conference, 2024, pp. 113–124.
[21] P. Lemey, M. Salemi, A.-M. Vandamme, The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing, Cambridge University Press, 2009.
[22] M. Zhu, A. I. Wang, Model-driven game development: A literature review, ACM Comput. Surv. 52 (6) (2020) 123:1–123:32. doi:10.1145/3365000.
[23] K. Empire, Kromaia on steam.
URL https://store.steampowered.com/app/285980/Kromaia/
[24] Wikimedia, List of playstation 4 games (a–l).
URL https://en.wikipedia.org/wiki/List_of_PlayStation_4_games_(A-L)
[25] B. Kim, Game mechanics, dynamics, and aesthetics, Library technology reports 51 (2) (2015) 17–19.
[26] D. Blasco, J. Font, M. Zamorano, C. Cetina, An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering, Journal of Systems and Software 171 (2021) 110804.
[27] M. Nei, Molecular Evolutionary Genetics, Columbia University Press, 1987.
[28] G. Weyenberg, R. Yoshida, Chapter 12 - reconstructing the phylogeny: Computational methods, in: R. S. Robeva (Ed.), Algebraic and Discrete Mathematical Methods for Modern Biology, Academic Press, Boston, 2015, pp. 293–319. doi:https://doi.org/10.1016/B978-0-12-801213-0.00012-5.
URL https://www.sciencedirect.com/science/article/pii/B9780128012130000125
[29] K. Holsinger, B. Weir, Genetics in geographically structured populations: Defining, estimating and interpreting fst, Nature reviews. Genetics 10 (2009) 639–50. doi:10.1038/nrg2611.
[30] D. Baum, S. Smith, Tree Thinking: An Introduction to Phylogenetic Biology, Macmillan Learning, 2012.
URL https://books.google.es/books?id=zW_ApwAACAAJ
[31] G. Weyenberg, R. Yoshida, Chapter 12-reconstructing the phylogeny: Computational methods. algebraic and discrete mathematical methods for modern biology. rs robeva (2015).
[32] V. I. Levenshtein, Binary codes capable of correcting spurious insertions and deletions of ones, Problems of Information Transmissions 1 (1) (1965) 8–17.
[33] N. Saitou, M. Nei, The neighbor-joining method: a new method for reconstructing phylogenetic trees, Mol Biol Evol 4 (4) (1987) 406–425.
[34] A. V. Brower, R. T. Schuh, Biological Systematics: Principles and Applications, Cornell University Press, 2021.
[35] E. T. Barr, M. Harman, Y. Jia, A. Marginean, J. Petke, Automated software transplantation, in: Proceedings of the 2015 International Symposium on Software Testing and Analysis, 2015, pp. 257–269.
[36] M. D. M. Zamorano López, D. Blasco, C. Cetina, F. Sarro, Video game procedural content generation through software transplantation, in: International Conference on Software Engineering: Software Engineering in Practice, IEEE/ACM, 2025.
[37] M. Farshbafnadi, S. Razi, N. Rezaei, Chapter 7 - transplantation, in: N. Rezaei (Ed.), Clinical Immunology, Academic Press, 2023, pp. 599–674. doi:https://doi.org/10.1016/B978-0-12-818006-8.00008-6.
URL https://www.sciencedirect.com/science/article/pii/B9780128180068000086
[38] C. Browne, F. Maire, Evolutionary game design, IEEE Transactions on Computational Intelligence and AI in Games 2 (1) (2010) 1–16.
[39] R. Gallotta, K. Arulkumaran, L. B. Soros, Evolving spaceships with a hybrid l-system constrained optimisation evolutionary algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 711–714. doi:10.1145/3520304.3528775.
URL https://doi.org/10.1145/3520304.3528775
[40] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.
[41] A. Vargha, H. D. Delaney, A critique and improvement of the cl common language effect size statistics of mcgraw and wong, Journal of Educational and Behavioral Statistics 25 (2) (2000) 101–132. arXiv:http://jeb.sagepub.com/content/25/2/101.full.pdf+html.
[42] N. Cliff, Dominance statistics: Ordinal analyses to answer ordinal questions., Psychological bulletin 114 (3) (1993) 494.
[43] N. Cliff, Ordinal methods for behavioral data analysis, Psychology Press, 2014.
[44] A. Arcuri, L. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Softw. Test. Verif. Reliab. 24 (3) (2014) 219–250. doi:10.1002/stvr.1486.
URL http://dx.doi.org/10.1002/stvr.1486
[45] J. Schell, The Art of Game Design: A book of lenses, CRC press, 2008.
[46] R. Hunicke, M. LeBlanc, R. Zubek, et al., 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, 2004, p. 1722.
[47] D. Blasco, J. Font, F. Pérez, C. Cetina, Procedural content improvement of game bosses with an evolutionary algorithm, Multimedia Tools and Applications 82 (7) (2023) 10277–10309.
[48] R. J. Grissom, J. J. Kim, "Effect sizes for research: A broad practical approach, Mahwah, NJ: Earlbaum, 2005.
[49] S. S. Mangiafico, Summary and analysis of extension program evaluation in r, Rutgers Cooperative Extension: New Brunswick, NJ, USA 125 (2016) 16–22.
[50] C. König, K. Rosiak, L. Cleophas, I. Schaefer, True variability shining through taxonomy mining, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 182–193. doi:10.1145/3579027.3608989.
URL https://doi.org/10.1145/3579027.3608989
[51] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.
[52] F. M. B. Boaventura, V. T. Sarinho, Mendiga: A minimal engine for digital games, Int. J. Comput. Games Technol. 2017 (2017) 9626710:1–9626710:13. doi:10.1155/2017/9626710.
[53] V. T. Sarinho, A. L. Apolinário, E. S. Almeida, A feature-based environment for digital games, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 518–523.
[54] D. Castro, C. Werner, Rebuilding games at runtime, IEEE, 2021, pp. 73–77. doi:10.1109/ASEW52652.2021.00025.
[55] C. Lima, I. do Carmo Machado, E. S. de Almeida, C. von Flach G. Chavez, Recovering the product line architecture of the apo-games, ACM, 2018, pp. 289–293. doi:10.1145/3233027.3236398.
[56] C. Lima, W. K. G. Assunção, J. Martinez, I. do Carmo Machado, C. von Flach G. Chavez, W. D. F. Mendonça, Towards an automated product line architecture recovery: The apo-games case study, ACM, 2018, pp. 33–42. doi:10.1145/3267183.3267187.
[57] C. Lima, C. Chavez, E. S. de Almeida, Investigating the recovery of product line architectures: An approach proposal, Springer International Publishing, Cham, 2017, pp. 201–207.
[58] J. Krüger, W. Fenske, T. Thüm, D. Aporius, G. Saake, T. Leich, Apogames: a case study for reverse engineering variability from cloned java variants, ACM, 2018, pp. 251–256. doi:10.1145/3233027.3236403.
[59] R. A. F. Moreira, W. K. Assunção, J. Martinez, E. Figueiredo, Open-source software product line extraction processes: the argouml-spl and phaser cases, Empirical Software Engineering 27 (4) (2022).
[60] J. Martinez, X. Tërnava, T. Ziadi, Software product line extraction from variability-rich systems: The robocode case study, SPLC '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 132–142. doi:10.1145/3233027.3233038.
[61] J. Debbiche, O. Lignell, J. Krüger, T. Berger, Migrating java-based apogames into a composition-based software product line, ACM, 2019, pp. 18:1–18:5. doi:10.1145/3336294.3342361.
[62] S. Apel, C. Kästner, C. Lengauer, Language-independent and automated software composition: The featurehouse experience, IEEE Transactions on Software Engineering 39 (1) (2013) 63–79. doi:10.1109/TSE.

2011.120.

[63] M. Preuss, A. Liapis, J. Togelius, Searching for good and diverse game levels, in: 2014 IEEE Conference on Computational Intelligence and Games, IEEE, 2014, pp. 1–8.

[64] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, G. N. Yannakakis, Procedural content generation through quality diversity, in: 2019 IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–8.

[65] A. S. Melotti, C. H. V. de Moraes, Evolving roguelike dungeons with deluged novelty search local competition, IEEE Transactions on Games 11 (2) (2018) 173–182.

[66] D. Blasco, A. Iglesias, J. Echeverría, F. Pérez, C. Cetina, Introducing phylogenetics in search-based software engineering: Phylogenetics-aware sbse, ACM Trans. Softw. Eng. Methodol.Just Accepted (Jan. 2025). doi:10.1145/3715002.
URL https://doi.org/10.1145/3715002

[67] S. Navarro, A. Iglesias, F. Pérez, C. Cetina, J. Font, Phylogenix: Bringing phylogenetics to unity, in: Proceedings of the 28th ACM International Systems and Software Product Line Conference, 2024, pp. 38–41.