Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Evaluación de los beneficios de las líneas de productos software en la ingeniería de software de videojuegos

Autor del proyecto: Jose Ignacio Trasobares Ibor

Director del proyecto: Lorena Arcega Rodríguez

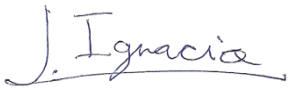
Zaragoza, 26 de junio de 2023

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma



Fecha

26 de junio de 2023

Dedicatoria y Agradecimiento

Quiero agradecer a toda mi familia por todo el apoyo que me ha dado, en especial a mis padres, que fueron los que más soportaron y les agradezco de haberme dado esta oportunidad sin ellos no hubiera podido estar entregando este proyecto. También a mi abuela que siempre estaré agradecido por haber estado ahí escuchando mis momentos malos y haberse preocupado por mí.

Agradecer a todos mis amigos que he conocido durante estos años por todos estos años compartidos y disfrutados, también por todo el apoyo

También querría agradecer a todo el grupo de investigación SVIT que desde el primer día me abrieron las puertas. En especial a Carlos Cetina por haber confiado en mi en todos los proyectos en los que he trabajado y en los venideros.

Y por último querría agradecer a mi directora Lorena, por todo el apoyo que me ha ido dando desde que empezó siendo una idea hasta en la redacción de este trabajo, sin su ayuda este trabajo no hubiera podido tener el reconocimiento que tiene.

Tabla de contenido

Resumen

La reutilización de código es una estrategia clave en el desarrollo de software. Dos técnicas que se basan en esta idea son Clone and Own (CaO) y las líneas de productos software (SPL). Estas técnicas buscan optimizar la eficiencia y la calidad al desarrollar software. Las SPLs han demostrado ser eficaces para reducir costes y tiempos, además de mejorar la calidad de los productos software desarrollados con esta técnica. Recientemente, ha crecido el interés en aplicar SPLs al desarrollo de videojuegos. Este trabajo evalúa si la adopción de una SPL en la ingeniería de software de videojuegos (GSE) puede generar los mismos beneficios que en la ingeniería de software clásico (CSE). En este trabajo se comparan estas dos técnicas, CaO y SPL utilizando el videojuego Kromaia. Se medirá la Corrección, Eficiencia y Satisfacción cuando los participantes desarrollan elementos de un videojuego comercial. Los resultados indican que los elementos desarrollados usando una SPL son más correctos que los desarrollados con CaO, pero no se ve una mejora significativa en Eficiencia o Satisfacción. Las conclusiones sugieren que las SPLs en GSE pueden desempeñar un papel diferente al que han desempeñado durante décadas en CSE.
Palabras Clave: Comparación empírica, Ingeniería de líneas de productos software, Ingeniería de software de videojuegos.

Abstract

Code reuse is a key strategy in software development. Two techniques based on this idea are Clone and Own (CaO) and software product lines (SPL). These techniques seek to optimize efficiency and quality when developing software. SPLs have proven to be effective in reducing costs and time, as well as improving the quality of software products developed with this technique. Recently, there has been a growing interest in applying SPLs to video game development. This paper evaluates whether the adoption of an SPL in video game software engineering (GSE) can generate the same benefits as in classic software engineering (CSE). This paper compares these two techniques, CaO and SPL, using the video game Kromaia. Correctness, Efficiency and Satisfaction will be measured when participants develop elements of a commercial video game. The results indicate that items developed using an SPL are more correct than those developed with CaO, but no significant improvement in Efficiency or Satisfaction is seen. The findings suggest that SPLs in GSE may play a different role than they have played for decades in CSE.
Keywords: Empirical comparison, Software product line engineering, Game software engineering.

1. Introducción

La reutilización de código es una práctica ampliamente utilizada en el campo de la ingeniería clásica de software (CSE, por sus siglas en ingles de *Classic Software Engineering*), ya que permite ser más eficiente al desarrollar nuevos productos software. Una de las técnicas más usadas es Clone and Own (CaO).

CaO [1, 2, 3, 4] se basa en realizar una reutilización oportunista de otros productos legados. Es decir, clonar parte del código de un producto software existente y adaptarlo al nuevo software sin ningún criterio especifico. Esto implica que el desarrollador tenga que identificar todos los bloques de código que crea que sean relevantes para el producto que está desarrollando. Por lo tanto, que esta técnica tenga efectos positivos depende mucho del conocimiento que tenga el desarrollador.

Para solucionar este problema, los investigadores desarrollaron las Líneas de Producto Software (SPLs, por sus siglas en ingles de *Software Product Lines*) [5, 6]. Según el Instituto de Ingeniería de software de la universidad de Carnegie Mellon, "Una línea de productos software es un conjunto de sistemas intensivos software que comparten un conjunto común y un gestor de características que satisfacen las necesidades específicas de un segmento de mercado o actividad concreta y que se desarrollan a partir de un conjunto común de activos básicos de una forma prescrita" [7]. Resumiendo, una SPL tiene como objetivo poder desarrollar varios productos software similares usando unas reglas preestablecidas para la elección de las características.

Además de solucionar el problema anterior, también las SPLs han demostrado ser eficaces en reducir costes y tiempos, además de mejorar la calidad en una gran variedad de tipos de software [8]. Por esa razón, hay investigaciones recientes que proponen aplicar las SPLs en el ámbito de los videojuegos [9, 10, 11]. Otra investigación reciente [12] aportó pruebas de que el desarrollo de videojuegos es diferente al desarrollo de software clásico. Uno de los ejemplos dados en esta investigación fue que los desarrolladores de videojuegos perciben más dificultades que los desarrolladores de software clásico a la hora de reutilizar código.

Hoy en día, la de los videojuegos es una de las industrias con mayor crecimiento en el mundo. Según un informe de 2019 [13], el número total de desarrolladores de software activos es de 18.9M. El mismo informe indica que la industria de los videojuegos hay 8.8M de desarrolladores activos. Esto significa que casi la mitad de los desarrolladores activos están trabajando en el sector de los videojuegos.

La mayoría de los videojuegos se desarrollan utilizando motores de videojuegos. Un motor de videojuegos es un entorno de desarrollo que integra dos motores (un motor para las físicas y otro para las gráficas) y un conjunto de herramientas para acelerar el desarrollo. Los más populares son Unity [14] y Unreal Engine [15], pero también hay estudios que han creado su propio motor (por ejemplo, CryEngine [16]).

Los modelos software [17, 18] representan una de las partes principales de los motores de videojuegos. Los modelos de software elevan el nivel de abstracción utilizando términos mucho más cercanos al dominio del problema. Esto significa que los desarrolladores pueden centrarse en el contenido del juego en sí, evitando los detalles de implementar físicas y gráficos. Los motores de videojuegos permiten a los desarrolladores crear contenidos directamente utilizando código (por ejemplo, C++) o modelos de software. Aunque Unity y Unreal Engine proponen su propio lenguaje de modelado, una encuesta en la revista "ACM Computing Surveys" [19] revela que los equipos que desarrollan software también adoptan los modelos UML [20, 18] y lenguajes de dominio específico (DSL, por sus siglas en ingles de *Domain Specific Language*).

Un DSL [21, 22] es un lenguaje diseñado para un dominio en específico. Al estar diseñado para usarse solo en un dominio, da ciertas ventajas como la facilidad de uso o la reducción de tiempos comparados al usar un lenguaje de propósito general (GPL, por sus siglas en ingles de *General Purpose Language*). Un GPL está diseñado para poderse usar en diferentes dominios, uno de los ejemplos más usados es UML [20, 18].

En este trabajo, se evalúa si la adopción de una SPL en la ingeniería de software para videojuegos (GSE, por sus siglas en inglés de Game Software Engineering) puede generar los mismos beneficios que en la CSE analizando el caso de estudio de un videojuego comercial: Kromaia. Kromaia ha sido analizado anteriormente por otros autores ya que utiliza un DSL para generar los jefes finales del videojuego [23, 24, 25, 26]. Se presenta un experimento en el que se comparan dos enfoques de desarrollo, CaO y SPL, ambos basados en el DSL de Kromaia, en términos de Corrección, Eficiencia y satisfacción. Un total de 28 participantes realizaron las tareas del experimento, desarrollando dos jefes de Kromaia.

Los resultados muestran que los jefes desarrollados con SPL son más correctos que los desarrollados con CaO, aunque no se ven cambios significativos en Eficiencia o satisfacción. Además, muestran que la eficiencia que había hecho atractivas las SPLs para CSE puede no serlo para GSE, eso implica que tal vez haya que replantearse el papel de las SPLs en GSE. A causa de estos resultados, se comentan las nuevas direcciones de investigación para las SPLs en GSE. En concreto, este trabajo revela que las SPLs en GSE pueden ser relevantes para generar nuevos

contenidos de videojuegos (uno de los temas candentes de la investigación en videojuegos) y balancear la dificultad (uno de los problemas fundamentales de los videojuegos).

El resto de la memoria se estructura de la siguiente forma: En la sección 2 se resumen los diferentes trabajos relacionados sobre este trabajo. En la sección 3 se explican los objetivos. En la sección 4 se presenta la metodología elegida. En la sección 5 se explican las diferentes técnicas de desarrollo software usadas en este trabajo. En la sección 6 se describe el experimento. En la sección 7 se muestran los resultados del experimento. En la sección 8 se desarrolla el estudio económico. Por último, en la sección 9 se describen las conclusiones de este trabajo.

2. Estado del Arte

Esta sección presenta los trabajos relacionados con este trabajo final de grado. Se divide en dos subsecciones teniendo en cuenta los temas tratados en este trabajo: Líneas de producto software aplicadas al dominio de los videojuegos, y evaluaciones o estudios de SPL y CaO.

2.1. Las SPL aplicadas al ámbito de los videojuegos

Algunas investigaciones se centran en la gestión de la variabilidad inherente a los videojuegos. La variabilidad es inherente a los videojuegos por el simple hecho de serlo.. Boaventura y Sarinho [27] presentan una colección de *game assets*[1] para crear pequeños videojuegos llamada Minimal Engine for Digital Games (MEnDiGa). MEnDiGa amplía un trabajo anterior llamado FEnDiGa [28] una plataforma de línea de productos que es capaz de integrar y adaptar características de juego representadas en diferentes tipos de motores de juegos disponibles. En MEnDiGa, desarrollan características lógicas y módulos para representar, interpretar y adaptar características del juego para su funcionalidad en múltiples plataformas de juego. Sus resultados muestran que el núcleo de los elementos de juego desarrollados con MEnDiGa puede ser independiente, reutilizable y realizado a gran escala. Castro y Werner [29] presentan un prototipo de juego desarrollado aplicando una Línea Dinámica de Producto Software (DSPL, por sus siglas en ingles de *Dynamic Software Product Line*) para generar modificaciones de juego de forma sistemática. Las DSPL se basan en una SPL, pero añaden una nueva característica: que son reconfigurables en tiempo de ejecución. El prototipo creado en la investigación demuestra la posibilidad de automatizar el proceso, teniendo una línea de productos donde el juego original es el núcleo de las funcionalidades del juego.

Siendo similar, Debbiche et al. [30] presentan una investigación sobre la creación de líneas de productos software en el área de los videojuegos. Para ello primero hicieron una extracción de todos los artefactos de 5 juegos multiplataforma de la familia Apo-Games. Una vez extraídos los artefactos, crean una SPL con la que intentan desarrollar 3 de los 5 juegos previamente mencionados. De estas pruebas concluyen que la extracción de una línea de productos software es complicada y requiere mucho tiempo. Recomiendan asegurarse de que la línea de productos software pueda probarse en todo momento para garantizar la correcta transformación de las

[1] *Game assets*: Elementos visuales, auditivos o de programación utilizados en un videojuego, como personajes, objetos, fondos, efectos de sonido o código fuente.

características . Por último, recomiendan adoptar las nuevas características de forma incremental para facilitar la extracción.

Otro ejemplo está en el mundo de los móviles. Nascimiento et al. [31] presentan un enfoque para implementar una SPL en los juegos para móviles. Para poder desarrollarla, establecen 3 fases: modelado, implementación y *testing*. Después de realizar estas fases describen 3 ejemplos de videojuegos de aventuras para móviles. Para finalizar concluyen que los resultados han sido positivos en el caso de implementar líneas de productos software en juegos para móviles y la facilidad de aplicar etiquetas de compilación condicional en el código. Pero también comenta que las etiquetas podrían ser un problema si la familia de productos crece demasiado.

Otras investigaciones se centran en la creación de SPLs aplicando reingeniería. Lima et al. [32, 10] presentan dos propuestas para el uso sistemático de técnicas de recuperación de la arquitectura de software (SAR, por sus siglas en ingles de S*oftware Architecture Recovery*). SAR es un método para recuperar la información de una arquitectura a un nivel bajo de representación. En este caso querían recuperar una arquitectura de línea de producto (PLA, por sus siglas en ingles de *Product Line Arquitecture*). Una PLA es una estructura o diseño arquitectónico utilizado de las líneas de productos software. En ambos su enfoque consiste en obtener el subconjunto mínimo de información de la arquitectura de productos cruzados para una buena recuperación de PLA. Para ello, primero seleccionan las variantes del código, luego extraen la información estructural, después preprocesan la información y la exportan. En la segunda propuesta añaden un paso, en el cual realizan la identificación con dos subtareas *Threshold Analysis*[2] y *Formal concept Analysis*[3]. Ya habiendo realizado el proceso, lo evalúan usando los proyectos de la familia Apo-Games para comprobar si el enfoque es el correcto. El resultado de este trabajo fue identificar diferentes valores atípicos y ayudar a tomar decisiones. La continuación del anterior trabajo [33] trata de identificar el subconjunto mínimo de la información de la arquitectura de productos cruzados para una recuperación eficaz de la PLA. Los resultados mostraron que eran capaces de identificar variantes de desviaciones atípicas y ayudar a los expertos a tomar decisiones.

Basándose también en Apo-Games, Krüger et al. [9] presentan una caracterización de las variantes de los juegos Apo-Games y reta a los investigadores a aplicar técnicas de ingeniería

[2] *Threshold Analysis* :Es un análisis para eliminar los valores atípicos

[3] *Formal concept Analysis* : Es un método para poder ver las relaciones y la estructura de los datos

inversa en modelos de características, localización de características, análisis de *smell code*[4], recuperación de arquitecturas y migración hacia una línea de productos software.

Con la conclusión del anterior trabajo muchos investigadores siguieron investigando en esa línea. Marchezan et al. [34] abordan el reto de identificar los fallos en el diseño e implementación de juegos clon usando el caso de Apo-Games. Para ello aplicaron cuatro puntos clave en 19 juegos de Apo-games. El primer punto es el sistema que consiste en localizar todas las propiedades (clases, métodos, atributos, llamadas o definiciones) de un código. Luego se aplican 6 Reglas de Consistencia para comprobar si contiene *smell code* o no. El siguiente punto que realizan es reparar. Este se basa en realizar los cambios necesarios para eliminar el *smell code* usando tres métodos: añadir, eliminar o modificar. Por último, crean un árbol de cada elemento modificado (método o clase) con todos los cambios que ha recibido. Los resultados mostraron que la media de *smell code* osciló entre 2.9 y 20.2 *smell codes* detectados. Su sistema de reparación pudo generar entre 4.9 y 29.89 reparaciones por *smell code* de media. Con esto pudieron concluir que el CaO puede llevar a la replicación de *smell code* en diferentes productos.

Otro ejemplo es el trabajo realizado por Åkesson et al. [35]. Presentan una investigación en el que su objetivo es la sistematización de las amenazas a la validez externa de los resultados experimentales de investigaciones de Líneas de Producto Software, para ello elige 4 juegos desarrollados para móvil de la familia Apo-games. Con ellos analizaron todos los componentes de esos juegos usando *Clone Detection* y *Feature Modeling* para desarrollar una SPL. Concluyeron dando 5 ideas para ser tratadas en un futuro: Disimilitud debido a librerías, esfuerzo inesperado de la marca, decisión sobre características, facilidad de lectura del código fuente y disponibilidad de las herramientas.

Moreira et al. [36] proporcionan y analizan datos empíricos de los procesos de extracción de dos casos de estudio de código abierto, ArgoUML-SPL y Phaser. En ambos casos se les ha realizado una transición de monolítico a SPL para poder entender más profundamente las extracciones e implicaciones de las SPLs. Su análisis se basó en el registro del control de versiones de los repositorios y las conversaciones de los desarrolladores durante el estudio. Aunque los resultados muestran que hay una gran diferencia entre el proceso de reingeniería de ArgoUML-SPL y Phaser, se encontraron problemas comunes en ambos casos. Estos problemas estaban relacionados con la falta de herramientas que llevaron a extracciones de características incompletas e inconsistentes, la complejidad en la gestión de las características pendientes de realizarse cuando

[4] *Smell Code* : Es cuando un código fuente parece indicar que tiene un problema profundo en él.

se utiliza el enfoque composicional y los problemas de no tener un modelo de variabilidad para la extracción.

De forma similar, Martínez et al. [37] trabajan en una experiencia relacionada con la creación de una SPL mediante la reingeniería de variantes del sistema, implementadas en torno a un juego educativo llamado Robocode, un juego para crear robots y realizar combates. Las tareas fueron realizadas por varios estudiantes en las que tenían varios ejercicios, con distintas dificultades para desarrollar la variabilidad del robot e implementar las características disponibles. Hablan de sus resultados desde distintos puntos de vista, como el valor educativo, el proceso de extracción, el análisis del tiempo y el esfuerzo necesarios. Sus resultados muestran que la implementación de las características fue la que más consumió tiempo, seguidas del análisis del dominio.

También se ha investigado el uso de las SPLs en motores de videojuegos. Martin Sierra et al. [11] hacen una comparativa entre dos motores de videojuegos (Unity y P5). Para ello analizan ambos motores, con el objetivo de saber cuál es el mejor motor para implementar una SPL. Para comparar, se usan 5 criterios de evaluación que son: Reutilización, montaje automático, rendimiento, curva de aprendizaje y licencia. Con estos criterios concluyen que la mejor plataforma sería Unity por su soporte, madurez y los elementos que facilitan el desarrollo de juegos con una alta complejidad. Aunque sea el elegido comentan ciertos problemas con Unity, tales como que no pudo resolver retos que P5 si pudo o que la curva de aprendizaje es más grande en Unity que en P5.

Otras investigaciones se basan en el hardware. Albassam et al. [38] exploran la aplicación de la tecnología SPL en el ámbito de los videojuegos aprovechando las diferencias entre las distintas plataformas de videojuegos para diseñar una PLA basada en componentes variables para un videojuego multiplataforma. Su enfoque trata de construir un modelo de dependencia de características para describir la variabilidad en los videojuegos multiplataforma. Para ello exploran la variabilidad en la interfaz de usuario, los dispositivos de entrada, los dispositivos de salida, la CPU, así como otras variabilidades en diversas plataformas de videojuegos. Después, diseñan una SPL basada en componentes variables que se adapta a cada videojuego de la línea de productos software. Para poder validar su enfoque, crearon dos versiones (Windows y Windows phone) de un juego de simulación usando una SPL. Esa SPL fue desarrollada con la técnica descrita en este trabajo.

Los trabajos anteriores aplican enfoques de SPL en el ámbito de los videojuegos. Sin embargo, estos trabajos no evalúan si los beneficios producidos por el uso de SPLs en CSE se aplican en GSE.

2.2. Evaluaciones o estudios de SPL y CaO

Algunos trabajos se centran en comparar diferentes técnicas de SPL. Constantino et al. [39] comparan dos herramientas de SPL centrándose en la facilidad de uso, los puntos fuertes y los puntos débiles. Sus resultados muestran que el principal problema de ambas herramientas es la falta de guías de usuario. Sin embargo, el análisis automatizado y el editor de modelos de características obtienen muy buenas puntuaciones. Dermerval et al. [40] compararon dos enfoques para el modelado de características basado en ontologías en el contexto de las DSPLs. Ellos compararon las ontologías usando el tiempo para realizar modificaciones, el impacto estructural al aplicar las modificaciones y la Corrección de los cambios aplicados. Sus resultados sugieren que las capacidades de razonamiento de las ontologías pueden apoyar eficazmente la reconfiguración de productos en el contexto de las DSPLs.

El desarrollo de software dirigido por modelos (MDD, por sus siglas en ingles de *Model driven Development*) es un método de trabajo que consiste en usar modelos como artefactos y desarrollar software con ello. En este contexto González-Huerta et al. [41] llevaron a cabo una familia de cuatro experimentos para evaluar un método de MDD para desarrollar SPLs. Usan un método para la derivación, evaluación y mejora de arquitecturas de software en el desarrollo de SPLs dirigido por modelos llamado QuaDAI. En sus experimentos utilizaron estudiantes de master y de grado de la rama de la informática, para realizar los experimentos. Utilizaron las variables de efectividad, eficiencia, percepción de la facilidad de uso (PEOU, por sus siglas en ingles de *Perceptions Ease of Use*), percepción de utilidad (PU, por sus siglas en ingles de *Perceived usefulness*) e intención a usarlo (ITU, por sus siglas en ingles de *Intention to Use*). Estas fueron usadas para comparar la estrategia de evaluación y mejora de QuaDAI frente al método de análisis de compensación de arquitecturas (ATAM, por sus siglas en ingles de *Architecture Tradeoff Analysis Method*), que es un proceso para mitigar riesgos y se aplica en los primeros ciclos del desarrollo. Los resultados muestran que el método basado en MDD obtiene los mejores resultados frente a ATAM.

Algunas de las investigaciones relacionadas con CaO se centran en las características. Ghabach et al. [3] propusieron un enfoque para apoyar la derivación de nuevas variantes de producto basadas en CaO. Su evaluación de un caso de estudio muestra que el soporte proporcionado puede reducir considerablemente el tiempo y los esfuerzos necesarios para lograr la derivación de un producto. De forma similar, Krüger et al. [42] propusieron un proceso semiautomático para identificar y mapear características entre sistemas heredados. Además de sugerir un enfoque de la visualización correspondiente y evaluar el proceso con un caso de estudio. Sus resultados

indican que el proceso puede utilizarse para mejorar la trazabilidad, preparar refactorizaciones y extraer SPLs.

Kehrer et al. [43] presentaron un proyecto llamado VariantSync para salvar la brecha entre CaO y SPLs combinando la mínima sobrecarga y flexibilidad de CaO con el manejo sistemático de la variabilidad en SPLs. Creen que VariantSync tiene un gran potencial para cambiar la forma en que los profesionales desarrollan sistemas de software multivariante.

Reinhartz-Berger y Sturm [44] estudian la comprensión de modelos de dominio especificados en ADOM. ADOM es un enfoque de modelado que está basado en un perfil UML compuesto por seis estereotipos. Para este trabajo realizaron un experimento con estudiantes universitarios que tenían que responder a preguntas de comprensión sobre un modelo de dominio que contaba con una guía de reutilización explícita y/o una especificación de variabilidad. Los resultados concluyen que la especificación explícita de la guía de reutilización dentro del modelo de dominio ayudó a los participantes a comprender el modelo. Además, la especificación explícita tanto de la guía de reutilización como de la variabilidad fue mejor que la especificación de la variabilidad sin guía de reutilización, pero fue peor que la especificación de la guía de reutilización sin variabilidad.

Bonifacio et al. [45] comparan un enfoque anotativo (PLUSS) y un enfoque compositivo (MSVCM) para especificar escenarios de casos de uso de líneas de productos utilizando una técnica multimodo para evaluar los beneficios. PLUSS es un método de modelado de dominio adaptado para el desarrollo de sistemas intensivos en software de larga duración y MSVCM que especifica la variabilidad y el conocimiento de configuración de una SPL y define un proceso para configurar la especificación de un producto. Llegan a la conclusión de que la especificación de una línea de productos utilizando MSVCM sólo requiere cambios localizados y requiere más tiempo para derivar las especificaciones de la línea de productos software.

Tizzei et al. [46] investigan el impacto de los componentes en la evolución de la SPL, su objetivo es valorar cómo los aspectos y los componentes promueven la estabilidad de las PLA en presencia de varios tipos de cambios evolutivos. Amplían su investigación evaluando la PLA frente al uso aislado de enfoques basados en componentes, orientado a objetos (OO, por sus siglas en ingles de *Object Oriented*) y orientado a aspectos (AO, por sus siglas en ingles de *Aspect Oriented*). OO es un paradigma en el que el desarrollo del software se basa en objetos que representan un objeto o idea y AO es paradigma en el que el desarrollo del software se basa en especificaciones. Para ello se realizó un análisis cuantitativo y cualitativo de la estabilidad de PLA y además se hicieron 4 implementaciones (AO, OO, componentes e híbrida). Para medir se usaron las métricas apropiadas para poder visualizar el impacto del cambio y la modularidad. Concluyeron que al

combinar aspectos y componentes se promueve una resistencia de la PLA mayor que el de las otras PLAs y apoyaba al diseño de PLAs de alta cohesión y débilmente acoplados.

Schilie et al. [47] desarrollan una nueva métrica y un procedimiento de análisis precisos. Comparan los sistemas de software en la medida de las sentencias individuales para recuperar la información sobre la variabilidad de los sistemas de software escritos en lenguajes de programación imperativos. Terminan desarrollando la métrica "*Fine-Grained Comparison Metric*". Esta métrica está compuesta por diferentes submétricas que evalúan un componente del código en específico. Además, crean una representación de familia de software de todos los sistemas analizados, denominada modelo 150%, que contiene artefactos de implementación y su información de variabilidad identificada. Para demostrar su eficacia, hicieron dos casos de estudio. Con todos esos datos concluyeron que su métrica muestra un buen rendimiento y que el modelo del 150% captura con precisión la información de variabilidad de los sistemas analizados.

Schultheiß et al. [1] analizan distintos trabajos que evalúan distintas variantes de CaO. Para ello informan sobre los trabajos en curso para investigar de forma más sistemática las amenazas a la validez externa de dichos resultados experimentales. Utilizando *n-way model matching*[5] como técnica representativa para apoyar la clonación y la posesión, concluyen en cuatro hipótesis que deben servir de base para la discusión y requieren pruebas más detalladas en futuros estudios.

Otros trabajos hacen evaluaciones empíricas añadiendo otros enfoques. Adam y Schimid [48] presentan un experimento en el que comparan dos enfoques de licitación de requisitos para las SPLs. Uno de ellos es ingeniería tradicional de requisitos de aplicación (ARE, por sus siglas en ingles de *Application Requirements Engineering*) y la otra es ARE integrado. Ambas se basan en la combinación de ingeniería de aplicaciones (AE, por sus siglas de *Application Engineering*) e ingeniería de requisitos (RE, por sus siglas de *Requirements Engineering*), pero entre ellas hay diferencias como las fases de la licitación o el apoyo a la obtención de requisitos específicos del cliente. Para poder comparar, realizaron un experimento con 26 estudiantes de ciencias de la computación. En el experimento tuvieron que realizar la extracción de las características para crear una SPL con uno de los dos métodos explicados anteriormente. Para los resultados se basaron en la puntuación que sacaron en los ejercicios y por último se propusieron 2 preguntas que tenían 4 hipótesis cada una. Ya habiendo analizado los resultados, sacaron la conclusión de que el ARE integrado es más preciso que el ARE tradicional, a causa de que ARE integrado

[5] *n-way model matching:* Es un método utilizado para comparar y encontrar similitudes entre múltiples modelos o conjuntos de datos de manera simultánea, teniendo en cuenta múltiples dimensiones o características.

distingue entre la reutilización de requisitos y la obtención adicional de requisitos específicos del cliente. También animan a otros investigadores a seguir investigando en esta área.

Figueiredo et al. [49] desarrollan un estudio cuantitativo que evoluciona dos SPLs para evaluar varias facetas de la estabilidad del diseño de sus implementaciones orientadas a aspectos. Para ello se evalúan dos técnicas de variabilidad: orientadas a aspectos (AO) y compilación condicional; dos son dos lenguajes de variabilidad. El diseño del experimento se estructura en 3 fases la primera es el diseño y la realización de escenarios de cambio de la SPL, luego la alineación de las versiones de la SPL, y por ultimo las evaluaciones cuantitativas y cualitativas de las versiones de la SPL y de los sucesivos lanzamientos. Para poder evaluarlo optaron por usar las métricas de atributos fundamentales de la modularidad, el acoplamiento, la cohesión y la concisión. Ya habiendo realizado el experimento, concluyeron que AO era mucho más estable que la compilación condicional, pero AO tiene problemas en la introducción de características obligatorias de amplio alcance, o, cuando se cambia una característica obligatoria por alternativas.

Tras analizar los trabajos anteriores, se vio la falta de estudios empíricos que comparen SPL y CaO. Sólo se pudo encontrar un trabajo que realizara esta comparación [50]. Los autores comparan las actuaciones de los ingenieros de software en el proceso de desarrollo de productos software utilizando SPL y CaO en un contexto industrial. Para comparar, los participantes realizaban los ejercicios aplicando SPL y CaO. Sus resultados concentran mejores valores de eficacia, eficiencia y satisfacción con el enfoque SPL. En contraste el presente trabajo, se centraron en CSE y no abordaron GSE.

3.Objetivos

En esta sección se establecen los objetivos que servirán de guía en el desarrollo de este trabajo, estos objetivos son los mismos que se marcaron en la propuesta (disponible en el Anexo I: Propuesta del Proyecto). Los objetivos establecidos para este trabajo son los siguientes:

1. **Conocer el estado del arte de las SPL y de GSE:** Con el propósito de analizar la situación actual y la importancia de las SPLs y su combinación con GSE. Se realizará una revisión de la literatura científica (libros, artículos y conferencias relevantes) en el ámbito de estas áreas. De esta forma, se obtendrá una comprensión de los conceptos y aplicaciones utilizados en estos campos.
2. **Diseñar y realizar una evaluación para estudiar la utilización de las SPL en GSE:** Se diseñará y llevará a cabo un experimento para saber si las SPLs son eficaces en GSE. Se utilizarán métricas y técnicas de evaluación para comparar los resultados obtenidos con el uso de SPL y el uso de CaO. Para ello, los participantes tendrán que realizar una serie de ejercicios y cuestionarios con el fin de recopilar datos relevantes.
3. **Analizar los resultados de la evaluación para comprender el impacto que pueden tener las SPL en GSE:** Se realizará un análisis exhaustivo de los resultados obtenidos en la evaluación, incluyendo las ideas y comentarios surgidos de los participantes. El objetivo es comprender el impacto de las SPL en la GSE. Se examinarán las métricas, datos, opiniones y percepciones recopiladas durante la evaluación para identificar patrones, tendencias y posibles mejoras.

4. Metodología

Para este trabajo final de grado, al ser este un estudio empírico, se ha aplicado una de las metodologías propuestas por R. Wieringa [51] en su tesis doctoral. Esta metodología se basa en la idea del ciclo empírico, que está formada por cinco fases:

1. **Análisis del problema a investigar:** En esta fase se busca analizar el tema que se quiere tratar haciendo una lectura de las investigaciones previas de esta área y se definen las preguntas que se van a responder en este estudio. En este trabajo se buscan investigaciones que hablen de aplicaciones de SPL en videojuegos y estudios empíricos sobre SPL en CSE, esta fase esta desarrollada en la sección 2.
2. **Diseño de la investigación:** Para poder contestar a las preguntas de la fase anterior se prepara un diseño de los experimentos que se van a ejecutar, en ellas se definen como y cuantas tareas se van a realizar, cuantos participantes se van a necesitar y como se va a distribuir el tiempo en cada una de las fases del experimento, esta explicado en la sección 6.
3. **Validación de la investigación:** En esta fase se presenta una lista de elementos para poder dar validez a la investigación en curso. En este trabajo se describe un análisis de las posibles amenazas a su validez, ya sea en términos de construcción, internas, de conclusión o externas. Para garantizar la fiabilidad de los resultados obtenidos, se ha procurado minimizar o mitigar el impacto de dichas amenazas desde la propia planificación y diseño de los experimentos, esto es tratado en la sección 6.10.
4. **Ejecución de la investigación:** En esta fase es en el momento en el que se realizan los experimento que fueron diseñados en el paso dos de este ciclo, esto es tratado entre las secciones 6 y 7.
5. **Análisis de los resultados:** Finalmente, ya habiendo realizado el paso anterior se analizan los resultados obtenidos y se contestan a las preguntas que se han definido en el primer paso, con todo ello se sacan conclusiones y se marca una dirección para futuras investigaciones sobre esta área. En este trabajo es desarrollado en las secciones 7 y 8 en los que se analizan los resultados del experimento que se han realizado y se contestan a las preguntas que se redactaron en la primera fase. Por último, se sacan conclusiones relacionados con los resultados y se destacan ciertas ideas para nuevos trabajos.

A parte de las fases que tiene este ciclo, este ciclo incluye también un conjunto de preguntas que ayudan a desarrollar y ejecutar la investigación que se está realizando. En la Figura 1 se puede ver la relación de las fases con las preguntas sugeridas en la tesis de Wieringa.

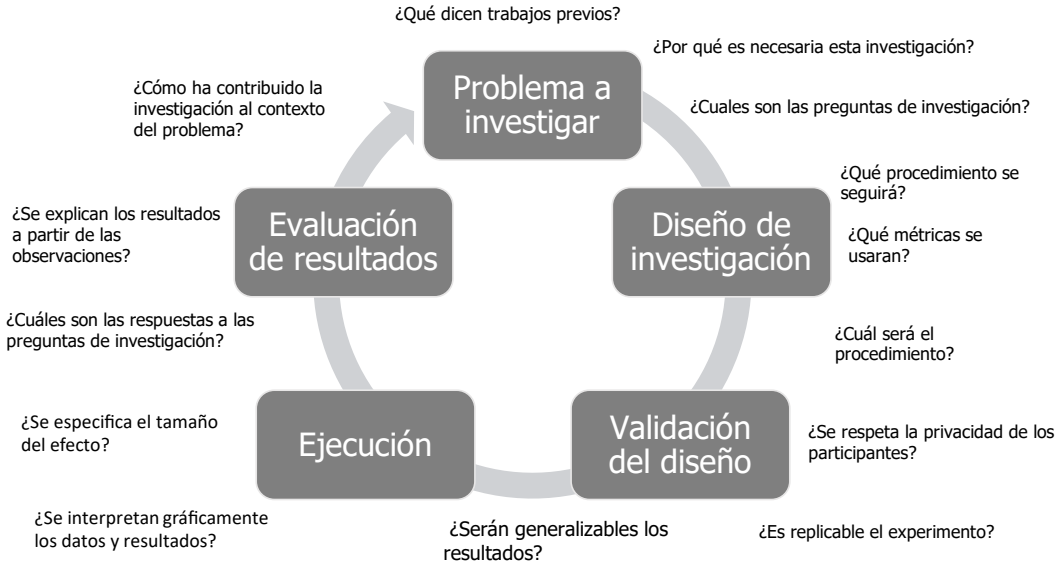


Figura 1: Esquema del desarrollo de la metodología

5. Background

Antes de llevar a cabo el experimento, es necesario analizar los elementos involucrados en el que se desarrollará el experimento. En esta sección del trabajo, se presentarán diferentes técnicas o elementos que se van a abordar y usar en este experimento.

5.1. Lenguaje de Dominio Especifico

El lenguaje de dominio específico [21, 22] (DSL, por sus siglas en inglés de *Domain Specific Language*) es un tipo de lenguaje de modelado diseñado para abordar problemas y tareas específicas en un dominio particular. A diferencia de los lenguajes de propósito general, como UML [20, 18], que son utilizados de forma generalizada para modelar software, un DSL se enfoca en un ámbito más limitado y ofrece un conjunto de abstracciones y herramientas especializadas para ese dominio en particular.

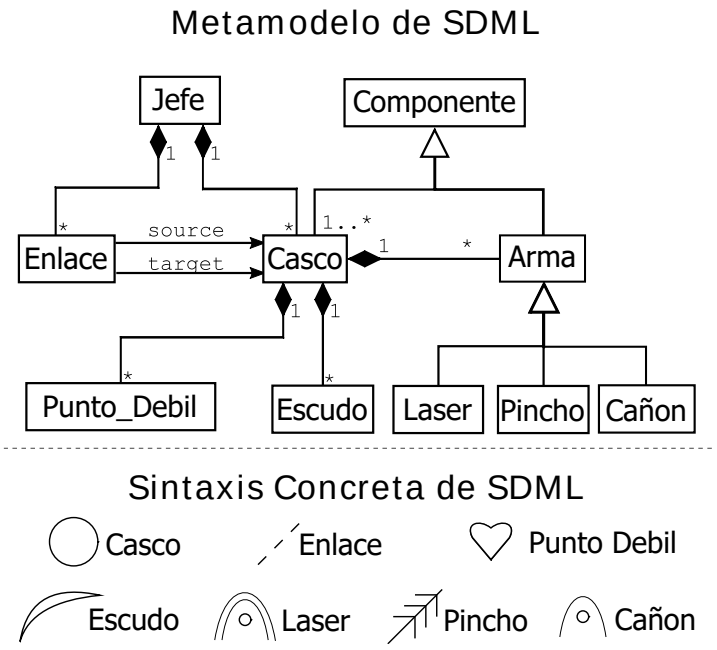


Figura 2: Metamodelo y sintaxis concreta de SDML

Los lenguajes de dominio específico se utilizan para describir o modelar conceptos específicos dentro de un dominio, lo que permite a los desarrolladores expresar soluciones de manera más concisa y comprensible. Al diseñar un DSL, se busca maximizar la legibilidad y la expresividad del código, lo que a menudo lleva a un mayor nivel de abstracción y a una reducción de la

complejidad. Para este trabajo se eligió un videojuego que hubiera sido desarrollado con un DSL como es el caso de videojuego Kromaia[6] .

Kromaia es un videojuego de tipo *shooter*[7] y mundo abierto. En este videojuego, el combate espacial es la característica principal. Durante una partida, el jugador tiene que enfrentarse diferentes tipos de oleadas de enemigos y jefes finales. El juego ofrece personalización de la nave, gráficos detallados, física realista y una atmósfera inmersiva.

Figura 3: Portada del juego Kromaia. Fuente "Kraken Empire".

El DSL que se utilizó para desarrollar Kromaia fue *Shooter Definition Modeling Language* (SDML). SDML define los diferentes componentes para poder desarrollar las diferentes entidades del videojuego. Entre ellas se define, la estructura anatómica (incluyendo qué partes se utilizan en ella, sus propiedades físicas y cómo están conectadas), la cantidad y distribución de partes vulnerables, armas y defensas en la estructura/cuerpo del personaje, además de los comportamientos de movimiento asociados a todo el cuerpo o a sus partes. Este lenguaje de modelado tiene conceptos como cascos, enlaces, puntos débiles y armas.

La Figura 2 muestra una versión simplificada del metamodelo SDML y su sintaxis concreta. El metamodelo completo contiene más de 20 conceptos, más de 20 relaciones y más de 60 propiedades [24]. Sin embargo, esta versión simplificada es lo suficientemente completa como para comprender los elementos que componen la estructura de un jefe y sus relaciones entre sí. En general, un jefe se compone de cascos y enlaces. Cada enlace une dos cascos. Un casco

[6] Kromaia Omega - Launch | PlayStation 4: https://youtu.be/EhsejJBp8Go

[7] *Shooter:* Es un género de los videojuegos en el que controlas a un personaje armado que dispara a enemigos utilizando armas de fuego en una perspectiva en primera persona.

puede contener puntos débiles que el jugador debe atacar, armas como cañones, láseres o pinchos, y escudos que protegen los puntos débiles y las armas.

En la Figura 4 se presenta un ejemplo real de jefe final de Kromaia. La Serpiente es el jefe final que el jugador debe derrotar para completar el Nivel 1. La parte inferior de la figura muestra el modelo del jefe utilizando la sintaxis concreta del SDML. Este jefe está compuesto por una serie de nueve cascos enlazados en línea. El primer casco es la cabeza, que incluye tres puntos débiles

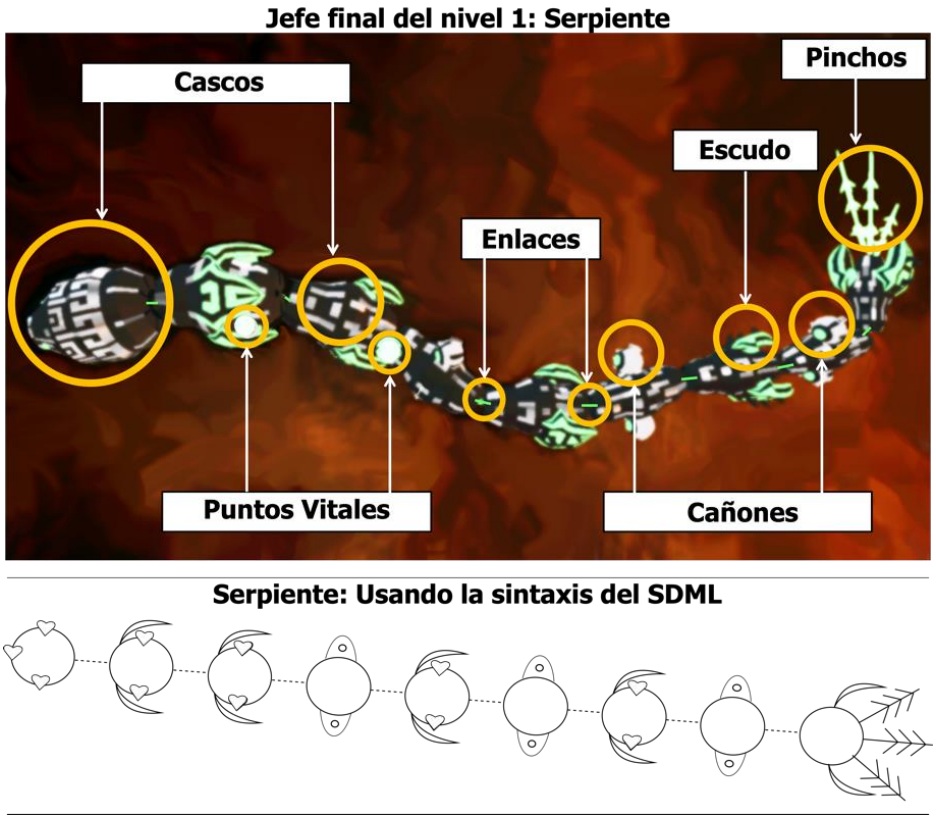


Figura 4: Ejemplo de un jefe final de Kromaia y su modelo en SDML

y se encargara del movimiento del jefe. Los siete cascos siguientes contienen dos puntos débiles cubiertos con escudos o con dos cañones. El último casco es la cola, que contiene dos escudos y tres pinchos que se usan para golpear al jugador.

5.2. Clone and Own

Clone and Own (CaO) [1, 2, 3, 4] es una práctica ampliamente utilizada en el desarrollo del software. Esta técnica trata de reutilizar el código de productos software existentes y modificarlos posteriormente por parte de los ingenieros de software encargados del desarrollo del nuevo software. Los cambios en el código se realizan con el objetivo de que el nuevo código mantenga

las características principales del código original, pero que se ajuste a las nuevas características especificadas por el cliente o el desarrollador.

Al utilizar el enfoque de CaO, los ingenieros de software pueden aprovechar implementaciones previas para desarrollar nuevos productos de manera más eficiente y rápida. La reutilización de código da varios beneficios, entre ellos, una mejora en el control de versiones del proyecto [52], ya que los ingenieros están familiarizados con el código desde el inicio. Además, ayuda a mantener una consistencia en el código entre los diferentes productos de una familia de software.

En el contexto de este trabajo, se aplicará CaO para desarrollar los jefes de Kromaia. El desarrollador cuenta con un conjunto de modelos SDML que han sido creados en el pasado, que les servirán como referencia. Estos modelos servirán como base para el nuevo jefe que se está creando. El desarrollador tiene la opción de reutilizar partes de los modelos existentes o utilizar uno de ellos como punto de partida para construir el nuevo jefe.

Por ejemplo, el desarrollador podría tomar el modelo de la serpiente existente y añadirle nuevas armas a uno de los cascos ya diseñados. También sería posible combinar los dos últimos cascos y conectarlos para crear un jefe serpiente con dos colas. La flexibilidad que ofrece la técnica CaO permite al desarrollador adaptar el código existente para cumplir con los requisitos específicos del nuevo jefe de Kromaia, al tiempo que mantiene la coherencia con el estilo de los jefes ya existentes.

5.3. Líneas de Producto Software

Una línea de productos software (SPL, por sus siglas ingles de *Software Product Lines*) es un enfoque de desarrollo de software que se basa en la creación de un conjunto de sistemas intensivos en software que comparten un conjunto común y un gestor de características, además satisface las necesidades específicas de un segmento de mercado o de una actividad concreta y se desarrolla a partir de un conjunto común de activos básicos de una forma prescrita [7].

El objetivo principal de una SPL es permitir el desarrollo de varios productos software similares utilizando reglas preestablecidas para la elección de las características. Para ello hay que definir un conjunto de características que puedan ser seleccionadas y configuradas de acuerdo con los requisitos y preferencias del cliente o del segmento de mercado objetivo.

La adopción de una SPL no solo resuelve problemas de desarrollo de software, sino que también ha demostrado ser eficaz en la reducción de costes y tiempos, así como en la mejora de la calidad en una amplia variedad de tipos de software [8]. Al tener una base común de activos y un gestor

de características centralizado, los equipos de desarrollo pueden reutilizar componentes existentes, lo que reduce el esfuerzo necesario para construir y mantener múltiples productos.

La Figura 5 muestra un ejemplo de la SPL para desarrollar jefes de Kromaia. La figura muestra uno de los modelos iniciales que pueden utilizarse para crear nuevos jefes. En la parte inferior muestra algunos modelos de características que representan la especificación de variabilidad y algunas de las características presentes en la biblioteca de la SPL. Cada característica corresponde a un fragmento de modelo que se expresa utilizando la sintaxis concreta de SDML.

Los elementos marcados con una P representan puntos de variación que pueden ser sustituidos por una característica, F, siguiendo la especificación de variabilidad. Por ejemplo, la especificación de variabilidad que se muestra en la primera posición P1 indica que puede sustituirse por los fragmentos de modelo F1 o F2. Además, algunas características pueden incluir puntos de variación que deben cumplirse con otra característica. Por ejemplo, la característica F2 tiene un punto de variación, P2, que debe sustituirse tras su especificación de variabilidad por F2, F3 o F4.

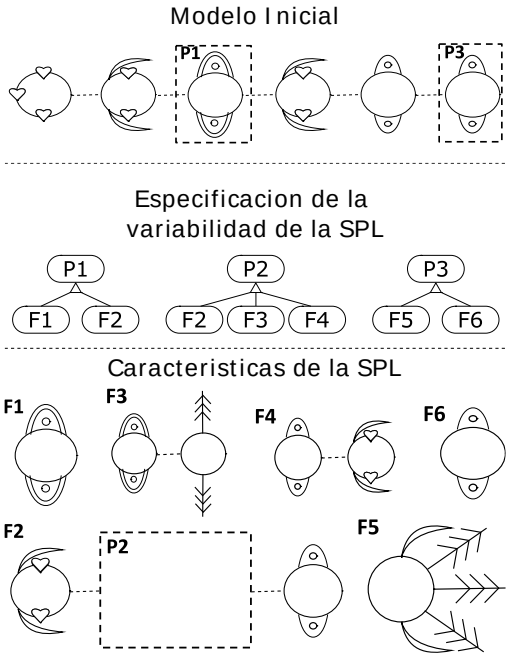


Figura 5: Ejemplo de una SPL para poder desarrollar jefes de Kromaia

6. Desarrollo

Esta sección explica el diseño del experimento. En esta sección se incluye la información de los objetivos del experimento, las variables estudiadas, las preguntas diseñadas para la investigación, el diseño elegido, los participantes, los artefactos utilizados durante el experimento, el procedimiento y el análisis estadístico elegido para el tratamiento de los datos obtenidos de los participantes.

6.1. Objetivos

De acuerdo con las directrices de Wohlin [53] para informar sobre experimentos de ingeniería de software, se han organizado los objetivos del experimento utilizando la plantilla *Goal Question Metric* para la definición de objetivos [54]:

El objetivo es analizar diferentes enfoques de desarrollo con fines comparativos, en lo que respecta a la Corrección, la Eficiencia y la Satisfacción del usuario de los modelos construidos, desde el punto de vista de estudiantes del grado de videojuegos y de desarrolladores expertos, en el contexto del desarrollo de videojuegos.

6.2. Variables

En este estudio, el factor investigado es el Enfoque del Desarrollo (DA, por sus siglas en ingles de *Development Approach*). Se investigan dos opciones: crear los jefes finales de un videojuego usando CaO o usando una SPL.

Para evaluar los efectos del uso de los diferentes enfoques, se selecciona la Corrección y la Eficiencia (relacionadas con el rendimiento de los participantes) como variables dependientes objetivas, y la Satisfacción del usuario (relacionada con la percepción de los participantes) como variable dependiente subjetiva.

Se mide la Corrección utilizando una rubrica, que se aplicó a los jefes desarrollados por los participantes después del experimento. El valor de esta métrica está en un intervalo de 0 a 100 y representa el porcentaje de puntos obtenidos según la plantilla de Corrección.

Para calcular la Eficiencia, se mide el tiempo empleado por cada sujeto para terminar la tarea, utilizando las horas de inicio y fin de cada tarea. La Eficiencia es la relación entre la Corrección y el tiempo empleado (en minutos) para realizar una tarea.

Se mide la Satisfacción mediante un cuestionario de 5 puntos en escala Likert basado en el Modelo de Aceptación de la Tecnología (TAM, por sus siglas en ingles de *Technology Acceptance Model*) [55]. Se descomponen la Satisfacción en tres variables subjetivas dependientes:

- Facilidad de uso percibida (PEOU, por sus siglas en ingles de *Perceived Ease of Use*): el grado en que una persona cree que aprender y utilizar una DA concreta requeriría menos esfuerzo.
- Utilidad percibida (PU, por sus siglas en ingles de *Perceived Usefulness*): el grado en que una persona cree que el uso de un DA concreto aumentará su rendimiento.
- Intención de uso (ITU, por sus siglas en ingles de *Intention to Use*): el grado en que una persona tiene intención de utilizar un DA.

Cada una de estas variables corresponde a elementos específicos del cuestionario TAM. Luego se hace una media de las puntuaciones obtenidas en estos elementos para obtener el valor de cada variable.

6.3. Preguntas de investigación e hipótesis

Las preguntas de investigación y las hipótesis nulas que se formulan son las siguientes:

RQ1. ¿Influye el DA utilizado para crear software para videojuegos en la Corrección del software? La hipótesis nula correspondiente es $H_{0,\,C}$: La DA utilizada para crear software para videojuegos no influye en la Corrección.

RQ2. ¿Influye la DA utilizada para crear software para videojuegos en la Eficiencia de los desarrolladores? La hipótesis nula para la Eficiencia es $H_{0,\,E}$: El DA no tiene efecto sobre la Eficiencia.

RQ3. ¿Es diferente la Satisfacción del usuario cuando los desarrolladores utilizan diferentes DA para crear software para videojuegos? Para responder a esta pregunta, se formulan tres hipótesis basadas en las variables PEOU, PU e ITU con sus correspondientes hipótesis nulas:

- $H_{0,\,PEOU}$ - La DA no tiene efecto sobre PEOU.
- $H_{0,\,PU}$ - La DA no tiene efecto sobre la PU.
- $H_{0,\,ITU}$ - La DA no tiene efecto sobre la ITU.

Las hipótesis se formulan como hipótesis de dos colas, ya que no se han encontrado estudios empíricos que apoyen una dirección específica para el efecto en el ámbito de los videojuegos.

6.4. Diseño

Para este experimento, se utilizó un diseño factorial cruzado con dos periodos, usando tareas diferentes (T1 y T2) para cada periodo. Los participantes se dividen aleatoriamente en dos grupos (G1 y G2). En el primer periodo del experimento, todos los participantes realizan la primera tarea (T1), los participantes del G1 utilizan CaO y los participantes del G2 utilizan SPL. A continuación, en el segundo periodo del experimento, realizaran la siguiente tarea (T2), pero ahora los participantes del G1 lo realizan con la SPL mientras que los participantes en G2 lo realizan con CaO.

El diseño de medidas repetidas aumenta la sensibilidad del experimento [56]. La observación del mismo sujeto utilizando las dos opciones, controla las diferencias entre participantes y mejora la robustez del experimento respecto a la variación entre participantes. Al utilizar dos secuencias diferentes para cada grupo (G1 utiliza CaO primero y SPL después, G2 utiliza SPL primero y CaO después) y diferentes tareas, el diseño contrarresta algunos de los efectos que pueden ser causados por el orden en que se utilizan los enfoques de desarrollo (es decir, efecto de aprendizaje, fatiga).

Antes de llevar a cabo cada experimento, para verificar el diseño, se realiza un estudio piloto con dos participantes. El estudio piloto permitió estimar el tiempo necesario para realizar las tareas y rellenar los cuestionarios, detectar errores tipográficos y semánticos, probar los instrumentos utilizados para recoger los datos y llevar a cabo el experimento. Los dos participantes del estudio piloto no participaron en ninguna de las sesiones realizadas.

6.5. Roles

En el experimento que se llevará a cabo, se asignarán diferentes roles a los participantes para asegurar el buen desarrollo del estudio y la obtención de resultados válidos. Estos roles son los siguientes:

- **Diseñador de las tareas:** Será la persona encargada de crear las diferentes tareas que se usaran para el experimento. Su función será comprobar que las tareas tengan un nivel adecuado para los participantes y de ellas se puedan sacar resultados relevantes. También establecerá los criterios de evaluación para medir el rendimiento de los participantes. Yo mismo junto con otra persona, que también participó en el desarrollo del experimento, nos encargamos de realizar las tareas asignadas a este rol.
- **Experto en DSL y SPL:** Sera la persona encargada de en explicar el experimento y las técnicas que tendrán que usar a los participantes. Su objetivo es darles a los participantes una comprensión clara y completa de las DSLs y las SPLs, mediante explicaciones de

teoría, ejemplos prácticos y respondiendo a las preguntas que puedan surgir. Yo asumí este rol y me encargué de realizar las tareas asignadas a este rol.

- **Ayudante del experto:** Sera la persona de apoyo para el experto durante el experimento. Su función principal sería estar disponible para atender las dudas, inquietudes o problemas técnicos que puedan surgir durante la ejecución de las tareas. Esta persona proporcionaría orientación y asistencia inmediata, ofreciendo explicaciones claras y soluciones prácticas a medida que los participantes avanzan en el experimento. Dos personas del Grupo de Investigación SVIT se encargaron de realizar las tareas asignadas a este rol.
- **Evaluador de respuestas:** Sera la persona encargada de corregir los ejercicios realizados por los participantes. Su función será revisar y visualizar (en caso de SPL, ya que las soluciones de los participantes solamente señalan las características que han elegido) las soluciones propuestas y evaluar su calidad y precisión. El corrector de ejercicios seguirá los criterios establecidos por el diseñador de las tareas para asegurar una evaluación justa y coherente. Yo asumí este rol y me encargué de realizar las tareas asignadas a este rol.
- **Entrevistador del Focus Group:** Esta persona será responsable de facilitar y moderar un debate al final del experimento. El objetivo del *Focus Group* es recopilar opiniones y experiencias de los participantes sobre el uso de las DSLs y las SPLs al realizar los ejercicios del experimento. El responsable del *Focus Group* lo dirigirá y asegurará que se obtenga información valiosa para complementar los resultados cuantitativos del estudio. Yo asumí este rol y me encargué de realizar las tareas asignadas a este rol.
- **Analista de Datos Estadísticos:** Esta persona será la responsable de coger los resultados del experimento y aplicarle técnicas estadísticas para obtener conclusiones significativas. Su función principal sería organizar y analizar los datos recopilados de manera precisa y rigurosa. Utilizando métodos estadísticos adecuados, para identificar patrones, tendencias y relaciones en los datos, y para realizar pruebas de significancia para determinar la validez de los resultados obtenidos. Una persona doctora en ingeniería del software y experta en evaluaciones empíricas perteneciente al Grupo de Investigación SVIT asumió este rol y se encargó de realizar las tareas asignadas a este rol.

6.6. Participantes

Se seleccionaron a los participantes mediante un muestreo de conveniencia [53]. Se invitaron a 20 profesionales del modelado de software o el desarrollo de videojuegos para participar en el experimento. De esos 20, 13 decidieron participar y completaron el experimento. Un total de 28 participantes con distintos conocimientos sobre modelado y desarrollo de videojuegos realizaron

el experimento. También se invitó a 20 estudiantes universitarios de tercer curso (desarrolladores sin experiencia) que estaban realizando la asignatura de desarrollo de videojuegos para móviles del Grado de Desarrollo de Videojuegos de la Universidad San Jorge. De estos participantes, 16 decidieron participar y 15 completaron las tareas y los formularios. Para poder realizar el experimento fue validado por el comité de ética de la universidad San Jorge, para que la universidad se permitiera realizar el experimento (Anexo III: Documentación del Experimento).

6.7. Objetos experimentales

Las tareas del experimento consistían en desarrollar dos jefes finales de Kromaia a partir de un *gameplay*[8] de Kromaia que mostraba su estructura y comportamiento.

Los participantes tuvieron acceso a material de entrenamiento con ejemplos de cómo crear un jefe utilizando CaO o la SPL.

Para la recogida de datos, se preparó un formulario con las siguientes secciones:

1. Una declaración de consentimiento informado que los participantes debían revisar y aceptar voluntariamente, en la que se explicaba claramente en qué consistía el experimento y cuál sería el tratamiento dado a los datos personales.
2. Un cuestionario demográfico para caracterizar la muestra.
3. Un cuestionario específico para recoger las respuestas de los participantes durante el experimento (sus tareas, sus tiempos y sus respuestas al cuestionario de satisfacción).

Los objetos experimentales utilizados en este experimento (el material de formación, las tareas y los formularios utilizados para los cuestionarios), así como los resultados y el análisis estadístico, están disponibles en el Anexo III: Documentación del Experimento.

6.8. Procedimiento experimental

Este experimento se realizó en dos días diferentes. La primera sesión del experimento se realizó online mediante la aplicación Teams con los profesionales. La otra sesión del experimento se realizó de forma presencial en la Universidad San Jorge con el grupo de estudiantes. Todas las sesiones se programaron con una duración de una hora y 45 minutos y se llevaron a cabo siguiendo el procedimiento experimental en la Figura 6:

1. (~ 5 min) Un instructor explicó las partes de la sesión y aclaró que no se trataba de una prueba de sus capacidades.

[8] *Gameplay*: Es un video que muestra la partida de un determinado juego.

2. (~10 min) Los participantes recibieron un tutorial sobre cómo crear los jefes del videojuego utilizando CaO o SPL para crear dichos jefes bajo el paradigma estudiado en el experimento.
3. (~5 min) Los participantes recibieron instrucciones claras sobre cómo acceder al formulario del experimento y los artefactos necesarios para completar la tarea. Los participantes se dividieron aleatoriamente en dos grupos (G1 y G2) y cada grupo tuvo acceso al formulario correspondiente.
4. (~5 min) Los participantes accedieron al formulario, después leyeron y confirmaron que habían leído la información sobre el experimento, el tratamiento de datos de su información personal y el carácter voluntario de su participación antes de acceder a los cuestionarios y tareas del experimento.
5. (~5 min) Los participantes cumplimentaron un cuestionario demográfico.
6. (~30 min) Los participantes realizaron la primera tarea. Los participantes de G1 tenían que utilizar la técnica de CaO para crear un jefe específico del videojuego, y los participantes de G2 tenían que crear el mismo jefe, pero utilizando una SPL. Tras entregar sus soluciones, los participantes completaron un cuestionario de satisfacción sobre el enfoque utilizado para el desarrollo.
7. (~30 min) Los participantes realizaron la segunda tarea, que consiste en la creación de otro jefe. Esta vez, los participantes del G1 lo crearon utilizando una SPL, mientras que los participantes del G2 lo crearon utilizando la técnica de CaO. A continuación, los participantes completaron el cuestionario de satisfacción.
8. (~15 min) Los participantes participaron en un *Focus Group* sobre las tareas realizadas.

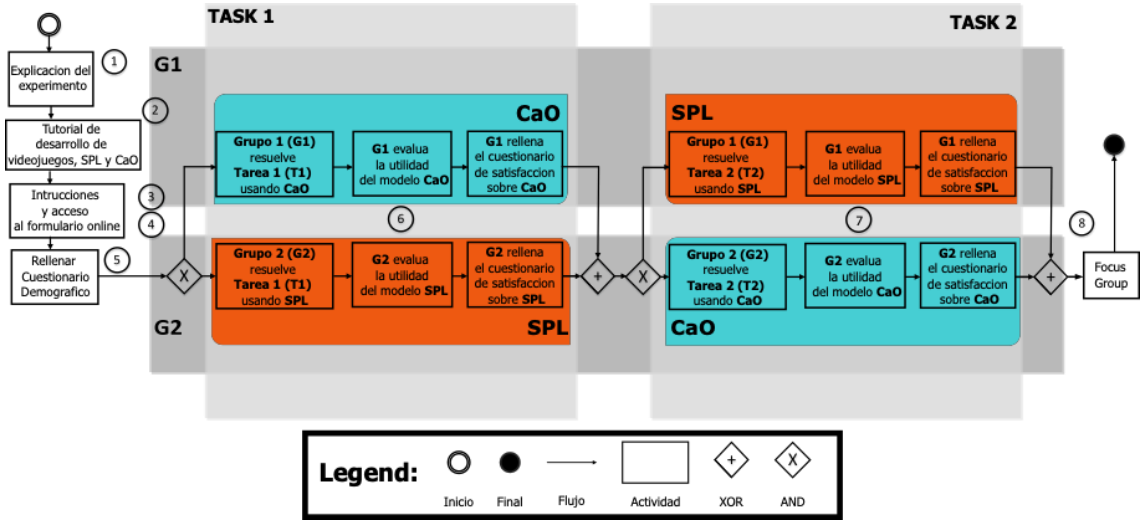


Figura 6: Esquema del procedimiento del experimento

6.9. Procedimiento de análisis

Para analizar los datos extraídos del experimento, se eligió el Modelo Lineal Mixto (LMM, por sus siglas en ingles de *Linear Mixed Model*) [57] para el análisis estadístico de los datos. El LMM maneja datos correlacionados resultantes de medidas repetidas, y permite estudiar los efectos de los factores que intervienen en un diseño cruzado (periodo y secuencia) y los efectos de otros factores de bloqueo (Experiencia) [56].

En la prueba de hipótesis, se aplicó la prueba de tipo III de efectos fijos con covarianza repetida no estructurada. El tipo III es la prueba por defecto, que permite al LMM producir los valores F y p exactos para cada variable dependiente y cada factor fijo. El supuesto para aplicar el LMM es la normalidad de los residuos de las variables dependientes. Para verificar esta normalidad, se usan las pruebas de Kolmogorov-Smirnov, así como inspecciones visuales del histograma y gráficos Q-Q normales.

En este estudio, la DA se definió como un factor fijo-repetido para identificar las diferencias entre utilizar CaO o SPL, y los participantes se definieron como un factor aleatorio (1|Subj.) para reflejar la variabilidad entre participantes. Las variables dependientes (DV, por sus siglas en ingles de *Dependent Variables*) de esta prueba fueron la Corrección, la Eficacia y las tres variables subjetivas relacionadas con la Satisfacción: PEOU, PU e ITU.

Para tener en cuenta los efectos potenciales de los factores que intervienen en un diseño cruzado a la hora de determinar el efecto principal del Enfoque de desarrollo (DA), se consideró que el Periodo y la Secuencia son efectos fijos. Con el fin de explorar los efectos de la experiencia del sujeto, se consideró la experiencia como factor fijo y sus combinaciones con otros factores fijos (DA, Paradigma, Secuencia y Periodo).

Se probaron diferentes modelos estadísticos para averiguar qué factores, además de la DA, podían explicar mejor los cambios en las variables dependientes. Algunos de estos modelos estadísticos se describen matemáticamente en la Fórmula (1). El modelo estadístico de partida (Modelo 0) refleja el factor principal utilizado en este experimento (DA) y el factor aleatorio Sujeto ($1|\ Subj.$). También se probaron otros modelos estadísticos que incluían otros factores fijos (Paradigma, Experiencia, Periodo o Secuencia), o sus combinaciones, que podrían tener efectos sobre las variables dependientes.

$$
\begin{aligned}
&(Model0)\ DV \sim DA + (1|Subj.)\\
&(Model\ 1)\ DV\ \sim DA + Experience + (1|Subj.)\\
&(Model\ 2)\ DV\ \sim DA + Experience + DA * Experience + (1|Subj.)\\
&(Model\ 3)\ DV\ \sim DA + Experience + Period + (1|Subj.)\\
&(Model\ 4)\ DV\ \sim DA + Sequence + Period + (1|Subj.)
\end{aligned}
\tag{1}
$$

El ajuste del modelo estadístico de los modelos probados se evaluó basándose en medidas de bondad de ajuste como el criterio de información de Akaike (AIC, por sus siglas de *Akaike's information criterion*) y el criterio de información bayesiano de Schwarz (BIC, por sus siglas en ingles de *Schwarz's Bayesian Information Criterion*). Se considera que el modelo con el menor AIC o BIC es el que mejor se ajusta [58, 26]. En el análisis, sólo se consideraron los modelos estadísticos que verifican la normalidad de los residuos de las variables dependientes. Por lo tanto, para describir los cambios en cada variable dependiente, se seleccionó el modelo estadístico que obtuvo el menor valor AIC o BIC de entre los modelos estadísticos que satisfacían la normalidad de los residuos.

Para cuantificar las diferencias en las variables dependientes debidas a factores fijos significativos, se calculó el valor d de Cohen [9] entre las alternativas de estos factores. Los valores de Cohen d entre 0.2 y 0.3 indican un efecto pequeño, los valores en torno a 0.5 indican un efecto medio y los valores superiores a 0.8 indican un efecto grande. Se seleccionaron histogramas y diagramas de caja para describir gráficamente los datos y los resultados.

6.10. Amenazas de validez

Para describir las amenazas a la validez de este trabajo, se ha utilizado la clasificación de Wohlin et al. [53]:

- **Validez de conclusión:** Se alcanza cuando existe una relación estadística entre el tratamiento y los resultados. La baja potencia estadística se minimizó porque el intervalo de confianza fue del 95%. Para minimizar la amenaza de pesca y de tasa de error, el análisis estadístico fue realizado por un investigador que no participó en el diseño de la tarea ni en el proceso de corrección. La amenaza de fiabilidad de las medidas se mitigó porque las medidas objetivas se obtuvieron a partir de los artefactos digitales generados por los participantes al realizar las tareas. La amenaza de fiabilidad de la aplicación del tratamiento se mitigó porque el procedimiento fue idéntico en cada una de las sesiones en las que se realizó el experimento.
- **Validez Interna:** Se consigue cuando las relaciones observadas entre el tratamiento y el resultado son relaciones causales y estas relaciones no son el resultado de un factor

sobre el que no tenemos control o que no hemos medido. Para evitar la amenaza de instrumentación, se realizó un estudio piloto para verificar el diseño y la instrumentación. La amenaza de maduración afectó al experimento, aunque las tareas se diseñaron con una complejidad similar, el efecto del periodo fue significativo para Eficiencia, PEOU e ITU. Los participantes fueron más eficientes en la segunda tarea, mientras que PEOU e ITU se valoraron mejor en la primera. Se minimizo esta amenaza utilizando un diseño cruzado para que el efecto afectara por igual a ambos enfoques evaluados. La amenaza de selección también afectó al experimento debido a la naturaleza voluntaria de la participación. Para evitar la desmotivación de los estudiantes, se seleccionaron aquellos estudiantes que estuvieran cursando asignaturas que cuyos contenidos se ajustaban al diseño del experimento. Las interacciones con la amenaza de selección afectaron al experimento porque había participantes que tenían distintos niveles de conocimiento del lenguaje de modelado y distintos niveles de conocimiento del dominio de los videojuegos. Para mitigar esta amenaza, el tratamiento se aplicó de forma aleatoria y los efectos del factor secuencia se han incluido en el análisis estadístico. Por otra parte, los participantes inexpertos están más representados en los resultados globales que los expertos.

- **Validez de construcción:** Se consigue cuando las medidas representan realmente lo que se está investigando en función de las preguntas de la investigación. El sesgo de monométodo se produce debido al uso de un único tipo de medida [59]. Para mitigar esta amenaza para las medidas de Corrección y Eficiencia, se mecanizan estas medidas en la medida de lo posible mediante plantillas de corrección. Se mitigan la amenaza a las medidas de Satisfacción utilizando un modelo ampliamente aplicado (TAM) [60]. La amenaza de adivinación de hipótesis aparece cuando el sujeto piensa en el objetivo y los resultados del experimento. Para mitigar esta amenaza, no se explican las preguntas de la investigación a los participantes. La amenaza de aprensión a la evaluación aparece cuando los participantes temen ser evaluados. Para debilitar esta amenaza, al principio del experimento, el instructor explicó a los participantes que el experimento no era una prueba de sus capacidades y, en el caso de los estudiantes, que ni su participación ni sus resultados afectarían a sus calificaciones en el curso. El sesgo de autor se produce cuando las personas implicadas en el proceso de creación de los artefactos del experimento influyen subjetivamente en los resultados. Para mitigar esta amenaza, las tareas se extrajeron de un videojuego comercial y fueron diseñadas por el mismo experto e investigador con una dificultad similar para todas las tareas. Estas personas no participaron en la ejecución del experimento.

- **Validez externa:** Se consigue cuando los resultados pueden generalizarse fuera del entorno del experimento. La interacción de la amenaza de selección y tratamiento es un efecto de tener un sujeto que no es representativo de la población que queremos generalizar. La participación de estudiantes en lugar de ingenieros de software no es un problema importante siempre que las preguntas de la investigación no se centren específicamente en desarrolladores profesionales [61], como es el caso de este experimento. Además, el factor experiencia se ha tenido en cuenta en el análisis de los datos. La interacción del entorno y la amenaza del tratamiento afectó al experimento. El *Focus Group* reveló que, para ambos enfoques y paradigmas de desarrollo, los participantes dedicaron mucho tiempo para evaluar si el jefe que se estaba creando era el jefe que querían desarrollar. Los participantes dijeron que pasaban mucho tiempo viendo el *gameplay* del videojuego para entender qué tenían que desarrollar. Esto no debería ser un problema en un escenario de desarrollo real porque se supone que los ingenieros de juegos entienden el juego que están creando. La amenaza de dominio se produjo porque el experimento se realizó en un dominio específico, como videojuegos y para un tipo de tarea muy concreta, es decir, desarrollar un jefe de videojuego a partir de Kromaia. La generalización de los resultados debe realizarse con cautela. Deberían realizarse otros experimentos en juegos diferentes para validar estas conclusiones.

7. Resultados

Esta sección incluye los resultados obtenidos a través del experimento, una discusión en la que se comentan las ideas dadas por los participantes y se interpretan los resultados obtenidos, se responden a las preguntas planteadas y se presentan las amenazas de validez del experimento. Por ultimo, se hace una revisión de los objetivos.

7.1. Resultados obtenidos

Los participantes completaron un formulario demográfico que se utilizó para caracterizar la muestra del experimento. La Tabla 1 muestra la cantidad de participantes en el experimento, divididos por su experiencia y mostrando las medias y desviación estándares de la edad, las horas diarias dedicadas al desarrollo de software (Tiempo de desarrollo) y las horas diarias dedicadas al trabajo con modelos (Tiempo de modelado). Se utilizó una escala Likert de 5 puntos para evaluar el conocimiento de los participantes sobre lenguajes de programación (Conocimientos de programación) y lenguajes de modelado (Conocimientos de modelado). También se muestra la media y la desviación estándar de las respuestas de los participantes.

	Todos los participantes	**Expertos**	**Inexpertos**
Número de participantes	28	13	15
Edad ± σ	25.8 ± 7.4	30.7 ± 8.4	21.5 ± 1.3
Tiempo de desarrollo ± σ	3.4 ± 2.5	4.5 ± 2.6	2.4 ± 2
Tiempo de modelaje ± σ	0.8 ± 1	1.2 ± 1.2	0.5 ± 0.6
Conocimiento de programación ± σ	3 ± 1.3	3.8 ± 1.3	2.3 ± 0.7
Conocimiento de modelaje ± σ	2.6 ± 1.6	3.5 ± 1.4	1.7 ± 1.2

Tabla 1: Resultados del cuestionario demográfico

La Tabla 2 proporciona los valores de la media y desviación típica de las variables dependientes: Corrección, Eficiencia, PEOU, PU e ITU para cada uno de los Enfoques de Desarrollo comparados (CaO y SPL). En todas las variables dependientes se encontraron diferencias significativas en las medias y desviaciones típicas en relación con el Enfoque de Desarrollo utilizado para crear un jefe

de un videojuego. Sin embargo, se puede observar que las diferencias observadas en Eficacia y Satisfacción son mínimas.

	Enfoque del Desarrollo	Experiencia		Periodo		Secuencia	
		Experto	Inexperto	Tarea 1	Tarea 2	G1 (CaO-SPL)	G2(SPL-CaO)
Corrección	CaO	51.87 ± 16.64	60.15 ± 12.43	44.7 ± 16.82	50.87 ± 18.39	53.03 ± 15.02	50.87 ± 18.39	53.03 ± 15.02
	SPL	64.13 ± 29.96	76.48 ± 22.98	53.43 ± 31.84	69.66 ± 21.88	59.34 ± 35.6	59.34 ± 35.6	69.66 ± 21.88
Eficiencia	CaO	3.41 ± 2.09	3.67 ± 1.57	3.19 ± 2.49	2.47 ± 1.37	4.5 ± 2.29	2.47 ± 1.37	4.5 ± 2.29
	SPL	3.29 ± 1.8	3.91 ± 1.83	2.76 ± 1.64	3.27 ± 1.34	3.31 ± 2.16	3.31 ± 2.16	3.27 ± 1.34
PEOU	CaO	3.54 ± 0.88	3.85 ± 0.97	3.28 ± 0.73	3.72 ± 0.67	3.33 ± 1.07	3.72 ± 0.67	3.33 ± 1.07
	SPL	3.78 ± 1.05	3.92 ± 1.2	3.67 ± 0.93	4.27 ± 0.63	3.37 ± 1.18	3.37 ± 1.18	4.27 ± 0.63
PU	CaO	3.8 ± 0.74	4.18 ± 0.71	3.5 ± 0.7	3.89 ± 0.69	3.7 ± 0.82	3.89 ± 0.69	3.7 ± 0.82
	SPL	3.7 ± 0.9	3.91 ± 1.01	3.48 ± 0.62	3.97 ± 0.78	3.47 ± 0.96	3.47 ± 0.96	3.97 ± 0.78
ITU	CaO	3.3 ± 1.14	3.92 ± 0.89	2.77 ± 1.08	3.5 ± 1.28	3.08 ± 0.95	3.5 ± 1.28	3.08 ± 0.95
	SPL	3.27 ± 1.26	3.65 ± 1.23	2.93 ± 1.22	3.85 ± 0.9	2.77 ± 1.33	2.77 ± 1.33	3.85 ± 0.9

Tabla 2: Valores de la media y la desviación típica ($\mu \pm \sigma$) de las variables dependientes, para el factor Enfoque de desarrollo en cada alternativa de los factores fijos.

Además, se presentan los efectos de los factores fijos considerados en el análisis estadístico:

- **Experiencia:** Con dos alternativas de participantes (Expertos e Inexpertos)
- **Periodo:** Con dos alternativas asociadas a las tareas (Tarea 1 y Tarea 2)
- **Secuencia:** Con dos alternativas que reflejan el orden en que los participantes utilizaron los enfoques de desarrollo: (G1: CaO-SPL, G2: SPL-CaO).

Para poder darle un valor a las diferencias en las variables dependientes debidas a cada factor, se analizaron los valores d de Cohen. La Tabla 3 muestra los valores d de Cohen de las variables dependientes para todos los factores fijos considerados en el análisis estadístico. Dependiendo del valor se dice que tiene un efecto de tamaño bajo (entre 0.2 y 0.3), medio (entre 0.3 y 0.5) o alto (más de 0.5), por debajo de 0.2 se dice que el valor es insignificante. Los valores que indican una variación media o alta debida al factor se destacan en negrita, y los que corresponden a diferencias significativas según las pruebas de hipótesis aparecen sombreados en gris. Los valores positivos indican diferencias a favor de la primera alternativa de los factores y los negativos a favor de la segunda alternativa. Según los valores d de Cohen de las variables dependientes para Enfoque de desarrollo, que se pueden ver en la primera columna de la Tabla 3, podemos decir que el factor Enfoque de desarrollo muestra un efecto medio a favor de SPL en cuanto a la variable Corrección, con un valor d de Cohen de -0.516. En cambio, el efecto es pequeño para PEOU, con un valor d de Cohen de -0.251 y el efecto del Enfoque de desarrollo en la Eficacia, PU e ITU es insignificante, con valores d de Cohen inferiores a 2 a favor de CaO.

	Enfoque del Desarrollo (CaO/SPL)	**Experiencia** (Expertos /Inexpertos)	**Periodo** (Tarea 1/Tarea 2)	**Secuencia** (G1 (CaO-SPL) /G2 (SPL-CaO))
Corrección	**-0.506**	**0.843**	0.127	-0.254
Eficiencia	0.062	**0.433**	**-0.544**	**-0.529**
PEOU	-0.251	**0.428**	**0.675**	-0.265
PU	0.119	**0.684**	**0.441**	-0.186
ITU	0.030	**0.853**	**0.658**	-0.279

Tabla 3: Valores d de Cohen de las variables independientes para cada factor fijo

En esa misma tabla, también se presentan los valores d de Cohen de las variables dependientes para los demás factores fijos considerados en el análisis estadístico. Estos valores indican que el factor Experiencia tiene efectos significativos en Corrección, PU e ITU a favor de los participantes expertos. Por otro lado, el factor Período muestra efectos moderados en la Eficacia, PEOU e ITU.

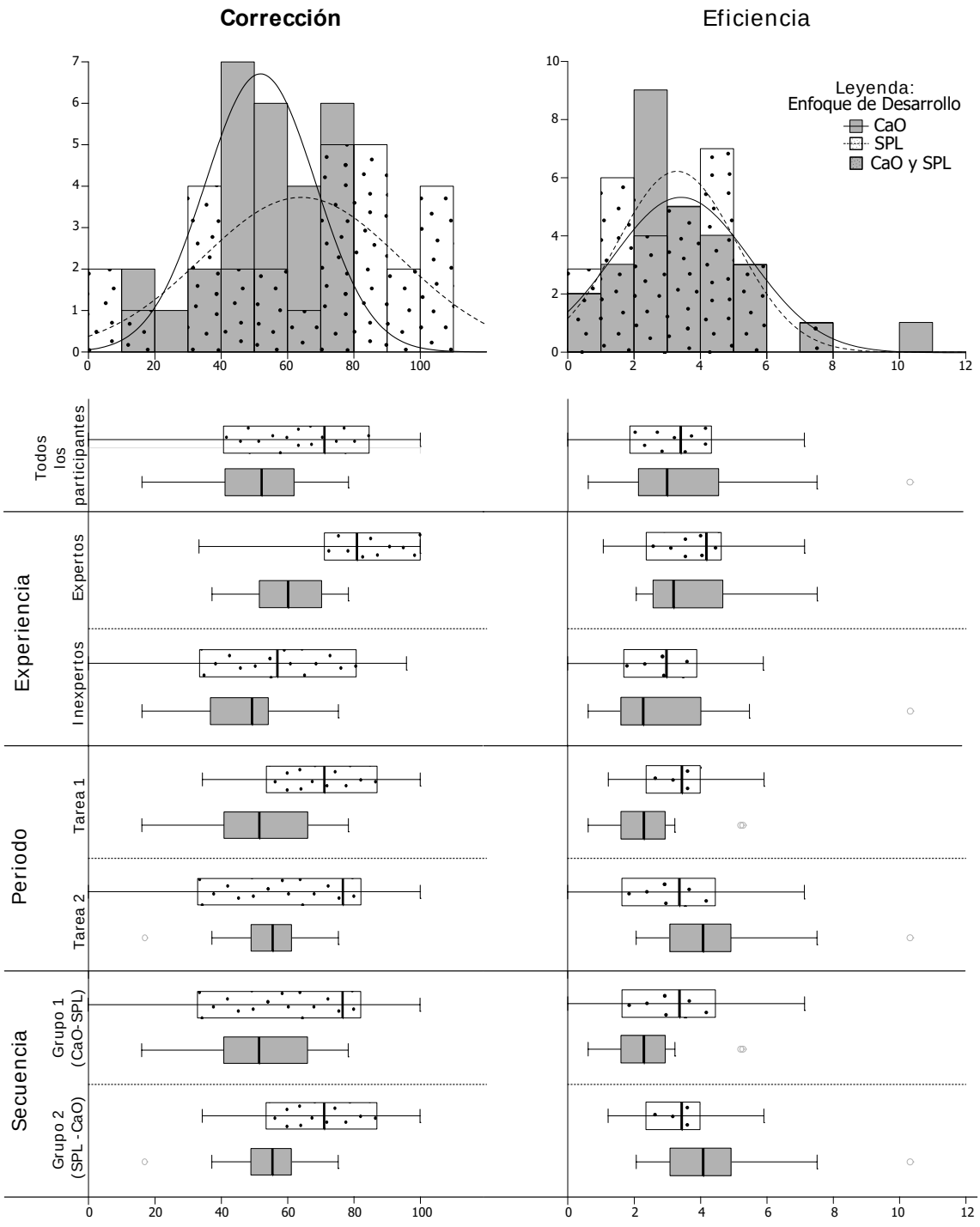


Figura 7: Histogramas con distribución normal y diagrama de cajas para la Corrección y Eficiencia

Los participantes fueron más eficientes en la segunda tarea, pero la primera tarea fue mejor valorada en PEOU e ITU. El factor Secuencia tiene un efecto pequeño en todas las variables dependientes, excepto en Eficiencia que el efecto es medio. Los participantes que empezaron a utilizar SPL en la primera tarea obtuvieron mejores resultados en ambas tareas que los que empezaron con CaO.

Los valores d de Cohen están relacionados con el porcentaje de no solapamiento entre las distribuciones de las variables dependientes para cada factor fijo. Los valores más altos corresponden a mayores porcentajes de no solapamiento y eso implica mayores diferencias. Los histogramas de la Figura 7 muestran las diferencias en Corrección en función del factor Enfoque de desarrollo. En los histogramas, las partes no solapadas tienen un único patrón (puntos o sombreado), mientras que las partes solapadas tienen ambos patrones. En el caso de la Corrección, las partes no solapadas se sitúan en torno al 20%, lo que corresponde a un efecto medio.

Los gráficos de caja de la Figura 7 correspondientes a todos los participantes muestran las diferencias en la Corrección debidas al enfoque de desarrollo. La mediana de la Corrección es un 20% mayor para SPL que para CaO, pero también aumenta la dispersión cuando los participantes utilizan el enfoque de la SPL. Los gráficos de caja de la Figura 7, correspondientes a participantes con y sin experiencia, muestran las diferencias en la Corrección debidas a la experiencia y al enfoque de desarrollo. Tanto los participantes expertos como los inexpertos desarrollan modelos más correctos cuando utilizan SPL en lugar de CaO. Sin embargo, independientemente del enfoque de desarrollo utilizado, los modelos creados por participantes expertos fueron más correctos que los desarrollados por inexpertos. En la misma grafica se puede ver el diagrama de cajas de Periodo y Secuencia. En Periodo la Tarea 1 ha sido un poco menos correcta que en la Tarea 2 en ambos enfoques, aunque prácticamente es mínima la diferencia. En Secuencia el Grupo 1 ha sido menos correcto que el Grupo 2.

La Figura 7 ilustra las diferencias de Eficiencia en función del factor Enfoque de desarrollo. Las partes no superpuestas para la Eficiencia son inferiores al 3%, lo que corresponde a un efecto insignificante entre el uso de CaO o SPL. Los gráficos de caja de la Figura 7 correspondientes a todos los participantes muestran diferencias insignificantes en la variable de Eficiencia debidas al método de desarrollo usado. Los gráficos de caja de la Figura 7 correspondientes a las alternativas de experiencia, muestran un resultado muy parecido a Corrección, ya que los expertos han sido más eficientes que los inexpertos en ambas alternativas (SPL y CaO). También las alternativas del periodo (Tarea 1 y Tarea 2) muestran las diferencias en Eficiencia debidas al enfoque de desarrollo y al periodo. Los participantes fueron más eficientes en la Tarea 2 que en la Tarea 1,

principalmente al utilizar el enfoque de desarrollo CaO. Por último, en Secuencia el Grupo 1 fue menos eficiente que el Grupo 2.

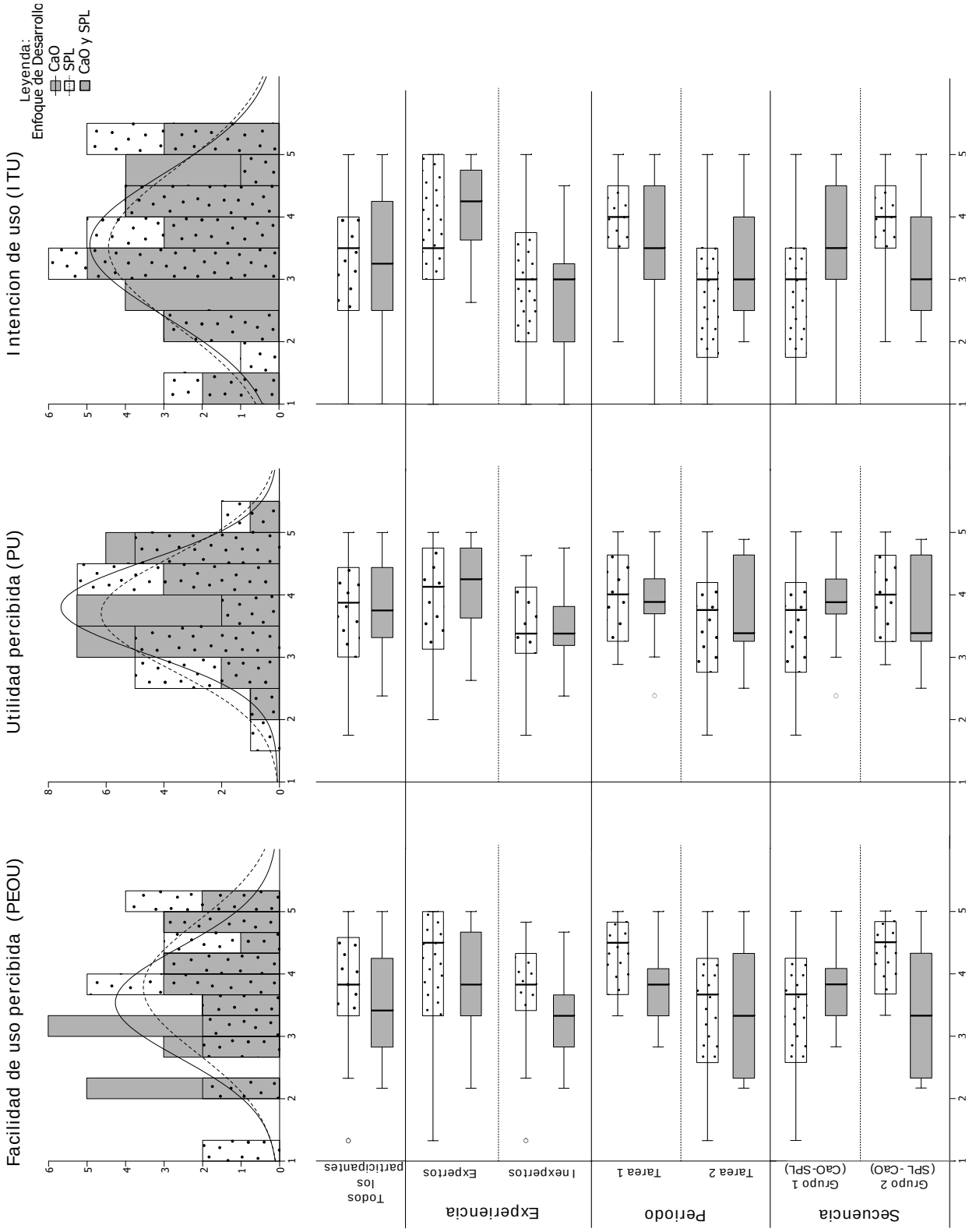


Figura 8: Histogramas con distribución normal y diagrama de cajas para la Satisfacción

La Figura 8 muestra las diferencias de las variables dependientes relacionadas con la Satisfacción (PEOU, PU e ITU). Como se puede observar, la parte no superpuesta en las 3 graficas es baja. Aunque PEOU tiene un porcentaje mayor (10%) de no superposición que las otras dos variables (PU 4.7% e ITU 1.2%), su tamaño de efecto es bajo. Tanto PU como ITU presentan un valor tan bajo que su efecto es insignificante. Teniendo en cuenta el factor Experiencia, se ve que en las 3 métricas los expertos han puntuado mejor que los inexpertos. Las grandes diferencias entre expertos e inexpertos se reflejan en los valores cohen d obteniendo un efecto medio para PEOU y un efecto alto para PU e ITU. Teniendo en cuenta el factor Periodo, la Tarea 1 obtuvo mejores resultados que la Tarea 2. De la misma forma que en el factor Experiencia, se ven grandes diferencias que concuerdan con el valor cohen d obteniendo un efecto medio en PU y alto en PEOU e ITU.

El Modelo Lineal Mixto utilizado para estudiar la significancia estadística de los cambios en las variables dependientes Corrección y Eficiencia, viene dada por esta fórmula:

$$DV \sim DA + Experience + Period + Sequence + DA * Experience + Experience * Period + Experience * Sequence + (1|Subj.) \quad (2)$$

Para estudiar la significancia estadística de los cambios en la Satisfacción (PEOU, PU, e ITU) el modelo estadístico elegido viene dada por esta fórmula:

$$DV \sim DA + Experience + Period + DA * Experience + DA * Period + Experience * Period + (1|Subj.) \quad (3)$$

Modelos como los de la fórmula (1), y otros LMM probados, no verificaron la normalidad de los residuos u obtuvieron valores más altos para las estadísticas de ajuste (AIC y BIC) que los obtenidos por los modelos seleccionados. Los factores y combinaciones de factores que forman parte de las fórmulas (2) y (3) explican los cambios en las variables dependientes. Sin embargo, según los resultados de las pruebas de hipótesis, no todos los cambios en las variables dependientes debidos a estos factores son significativos. La Tabla 4 muestra los resultados de la prueba de efectos fijos de tipo III para cada una de las variables dependientes y para cada factor fijo del modelo estadístico utilizado en cada caso. Los valores que indican diferencias significativas aparecen sombreados en gris.

	Enfoque del Desarrollo CaO/SPL	**Experiencia** Expertos/ Inexpertos	**Periodo** Tarea 1/ Tarea 2	**Secuencia** G1 (CaO-SPL)/ G2 (SPL-CaO)	**Enfoque del Desarrollo * Experiencia**	**Enfoque del Desarrollo * Periodo**	**Experiencia * Periodo**	**Experiencia * Secuencia**
Corrección	F=5.448 p=0.028	F=8.531 p=0.007	F=0.574 p=0.456	F=0.737 p=0.399	F=0.501 p=0.486	NA	F=0.031 p=0.863	F=4.34 p=0.048
Eficiencia	F=0.177 p=0.678	F=1.932 p=0.177	F=6.379 p=0.019	F=2.908 p=0.101	F=0.754 p=0.394	NA	F=0.37 p=0.549	F=0.632 p=0.435
PEOU	F=1.735 p=0.192	F=2.257 p=0.145	F=1.735 p=0.004	NA	F=0.572 p=0.452	F=0.885 p=0.356	F=0.216 p=0.643	NA
PU	F=0.17 p=0.684	F=6.139 p=0.02	F=3.026 p=0.095	NA	F=0.565 p=0.46	F=0.514 p=0.48	F=0.111 p=0.742	NA
ITU	F=0 p=0.993	F=11.899 p=0.002	F=6.293 p=0.019	NA	F=0.515 p=0.48	F=1.493 p=0.233	F=0 p=0.986	NA

Tabla 4: Resultados de la prueba de tipo III de efectos fijos para cada variable y factor. NA=No aplicable

7.2. Respuestas a las preguntas de investigación

Según los resultados obtenidos por la parte de Corrección, se encontró que el factor Enfoque de desarrollo tuvo un valor p inferior a 0.05, lo que lleva a rechazar la primera hipótesis nula y la respuesta a **RQ1 es afirmativa**. Esto indica que el enfoque de desarrollo utilizado para crear un jefe en un videojuego tiene un impacto significativo en la Corrección. Además, las pruebas de la hipótesis confirman la significancia estadística de las diferencias observadas entre las estadísticas descriptivas y las representaciones gráficas.

Los modelos realizados con SPL fueron más correctos que los realizados con CaO. También se encontró que el factor Experiencia y la combinación de Experiencia y Secuencia fueron estadísticamente significativos para explicar los cambios en la Corrección. Específicamente, los participantes expertos obtuvieron más modelos correctos que los inexpertos, tanto con CaO como con SPL. Además, las estadísticas descriptivas de las alternativas para la combinación de los factores Experiencia y Secuencia: Experto-G1 (μ = 71.87; σ = 20.52), Experto-G2 (μ = 64.17; σ = 19.22), Inexperto-G2 (μ = 49.064; σ = 25.50), e Inexperto-G1 (μ = 40.44; σ = 26.04) indican grandes diferencias entre los sujetos experimentados e inexpertos cuyo primer ejercicio se desarrolló utilizando CaO (Grupo 1). Se observan grandes diferencias entre los participantes expertos y los inexpertos en el Grupo 1, cuyo primer ejercicio se desarrolló utilizando CaO. Los participantes expertos del Grupo 1 obtuvieron los mejores resultados en Corrección en ambas tareas (CaO y SPL), mientras que los participantes inexpertos de este grupo realizaron los modelos menos correctos, también en ambas tareas.

En cuanto a la Eficiencia, el factor Enfoque de desarrollo obtuvo un valor p superior a 0.05, lo que nos lleva a no rechazar la segunda hipótesis nula y la respuesta **RQ2 es negativa**. Esto indica que el enfoque de desarrollo utilizado no tiene un impacto significativo en la Eficiencia. Sin embargo, se observó que los participantes dedicaron más tiempo al desarrollo cuando utilizaron SPL (μ = 20.6 minutos; σ = 5.72) en comparación con CaO (μ = 18.43 minutos; σ = 7.33), lo que contrarresta el efecto positivo del uso de SPL en la Corrección. Además, el periodo tuvo un impacto significativo en la Eficiencia, ya que los participantes fueron más eficientes en la segunda tarea, lo que podría indicar un efecto de aprendizaje.

En cuanto a las variables dependientes relacionadas con la satisfacción, el factor Enfoque de desarrollo obtuvo valores p superiores a 0.05, lo que nos lleva a no rechazar la hipótesis de que dicho factor tenga un impacto significativo en la satisfacción. Por lo tanto, la respuesta a la **RQ3 es negativa**. El enfoque de desarrollo no tiene un impacto significativo en la Satisfacción de los participantes al desarrollar un jefe de videojuego a partir de un videojuego. Sin embargo, se encontraron cambios significativos en la PEOU debido al periodo. Los participantes percibieron el

enfoque de desarrollo utilizado en la primera tarea, independientemente de si era CaO o SPL, como más fácil de usar que el enfoque de desarrollo utilizado en la segunda tarea. Además, la ITU fue más alta en la primera tarea que en la segunda. Además, el factor Experiencia tuvo efectos significativos y grandes sobre la satisfacción: los participantes expertos obtuvieron puntuaciones más altas que los inexpertos en PU e ITU para ambos enfoques utilizados en el experimento.

7.3. Discusión de los resultados

Según los informes registrados durante las dos últimas décadas por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon [7], Las SPLs proporcionan beneficios en Eficiencia y Corrección. Sin embargo, los resultados no muestran que las SPLs proporcionen diferencias significativas en Eficiencia en el ámbito de los videojuegos. Las diferencias existentes al trabajar entre CSE y GSE pueden significar que la Eficiencia no sea el punto fuerte para GSE. Para comprender mejor los resultados del experimento, se realizó un *Focus Group* con los participantes. El grupo tenía preguntas abiertas como las siguientes ¿Qué enfoque de desarrollo os ha gustado más y para qué utilizarías cada uno de ellos? ¿El proceso mental o los pasos que seguisteis para crear el jefe fueron los mismos con ambos enfoques?

El *focus group* reveló que, independientemente de si los participantes utilizaban CaO o SPL, el mayor tiempo usado para resolver los ejercicios fue al evaluar en ejecución si lo que estaban entendiendo era el jefe que querían desarrollar. Los participantes afirmaron que necesitaban comprobar constantemente el videojuego en tiempo real para poder comprender lo que tenían que desarrollar. Los beneficios de Eficiencia de la SPL no eran significativos porque el mayor esfuerzo correspondía a la parte de comprobar constantemente el videojuego en tiempo real y este punto es común tanto para CaO como para SPL. La mayoría de los participantes reconocen que la SPL no aceleró el proceso de desarrollo del videojuego.

A pesar de lo anterior, la mayoría de los participantes observaron beneficios de las SPLs para los videojuegos. En concreto, la mayoría de los participantes mencionaron que la SPL era muy relevante para crear nuevo contenido para videojuegos. Como explicaron los participantes, la creación de contenido es fundamental para el desarrollo de videojuegos. Los mundos de los videojuegos pueden necesitar una gran cantidad de contenido para poblarlos. Además, es habitual que, una vez lanzado el videojuego, se pueda seguir introduciendo nuevo contenido a través de actualizaciones que se conocen como contenidos descargables (también llamados DLC, por sus siglas en ingles de *Downloadable Content*).

Los participantes comentaron que pensar en términos de las características de la SPL les ayudaba a pensar en nuevo contenido para el videojuego. Ellos valoraron positivamente que combinando las características y siguiendo las reglas de la especificación de variabilidad, obtuvieron nuevos contenidos. Este nuevo contenido era efectivamente diferente pero similar a lo que ya existía en el videojuego, sin embargo, ellos dijeron que este tipo de contenido era relevante para desarrollar videojuegos. Además, los participantes imaginaron que este nuevo contenido derivado del ensamblaje de características puede ser útil como variantes de lo que ya se utilizaba en el videojuego para evitar la repetición. También podría utilizarse para rellenar partes secundarias del juego que estuvieran escasamente pobladas o incluso como fuente de inspiración para crear contenidos completamente nuevos.

En la comunidad de investigadores del ámbito de los videojuegos, la generación de contenidos para videojuegos es un tema candente. Muchos estudios [62, 63, 64, 65] hablan de cómo automatizar (total o parcialmente) la generación de contenidos. En concreto, la automatización de la generación de contenidos se conoce como Generación de contenido procedural (PCG, por sus siglas en ingles de *Procedural Content Generation*) [66] y utiliza las técnicas de Inteligencia Computacional más populares en la actualidad, como el aprendizaje automático y las basadas en búsquedas. Sin embargo, las SPLs no aparecen en ninguna de esos trabajos. Esto sugiere que el uso de SPLs para generar contenidos de videojuegos no ha sido explorado por la comunidad de investigadores en el ámbito de los videojuegos y puede ser una oportunidad para explorarlo en el futuro.

Además, se sugiere que las SPLs no tienen por qué verse únicamente como una alternativa a las técnicas actuales de PCG, ya que las SPL pueden ser un aliado cuando se combinan con las técnicas de PCG. Uno de los participantes recalcó que el punto fuerte de las SPLs en comparación con CaO era que en SPL sólo se necesitaban 10 decisiones, mientras que en CaO tenía que tomar entre 50 y 100 decisiones. Este espacio de búsqueda reducido de las SPLs puede ser más favorable para las técnicas de aprendizaje automático utilizadas en PCG para extraer patrones o para las técnicas basadas en la búsqueda utilizadas en PCG para explorar el espacio de búsqueda.

Por último, un grupo de participantes dijo que las SPLs pueden ser relevantes para ayudar a balancear la dificultad de los videojuegos. Balancear la dificultad de los videojuegos es uno de los principales problemas del desarrollo de videojuegos. El videojuego no puede ser demasiado difícil porque los jugadores se frustrarían, pero tampoco puede ser demasiado fácil porque los jugadores perderían el interés. Los participantes pensaron en cómo se podrían aumentar las características para incluir la idea de dificultad con el fin de obtener variantes del mismo contenido con diferentes grados de dificultad.

7.4. Revisión de los objetivos

Todos los objetivos presentados en la sección 3 han sido cubiertos:

1. **Conocer el estado del arte de las SPL y de GSE:** Se analizó la situación actual y la importancia de las SPLs y su combinación con GSE. La revisión de la literatura científica (libros, artículos y conferencias relevantes) en el ámbito de estas áreas se encuentra en la sección 2.
2. **Diseñar y realizar una evaluación para estudiar la utilización de las SPL en GSE:** Se diseñó y llevó a cabo un experimento para saber si las SPLs son eficaces en GSE. Se utilizaron métricas y técnicas de evaluación para comparar los resultados obtenidos con el uso de SPL y el uso de CaO. El diseño y la evaluación de este experimento se encuentra en las secciones 6.
3. **Analizar los resultados de la evaluación para comprender el impacto que pueden tener las SPL en GSE:** Se realizó un análisis exhaustivo de los resultados obtenidos en la evaluación, incluyendo las ideas y comentarios surgidos de los participantes. Los resultados, el análisis y la posterior discusión pueden encontrase en esta sección.

8. Estudio Económico

En esta Sección, se analizan todos los gastos asociados al desarrollo del estudio empírico. Estos gastos incluyen los recursos humanos necesarios, así como los costos de adquisición de equipos y software. Aparte de ello se evaluarán los beneficios económicos que podrían tener al usar una SPL.

8.1. Gastos del estudio empírico

En este estudio empírico podemos separar los gastos en 3 partes: Gastos de recursos humanos ,gastos de equipos y software y gastos de instalaciones.

- **Gastos de recursos humanos:** Durante el estudio empírico, se requirió la dedicación de un investigador durante un total de 240 horas. Además, se contó con la ayuda de una persona doctorada en ingeniería del software y experta en evaluaciones empíricas durante 20 horas para abordar aspectos específicos del análisis de los resultados.

  Según Glassdoor[9], el salario de un investigador es de 27,816 brutos euros anuales, que es lo mismo que 14.49 euros/hora, lo que resulta en un gasto total de:

  $$(240 + 20) horas * 14.49 \frac{\text{euros}}{\text{hora}} = 3{,}767.4\ euros$$

  El gasto total en recursos humanos es de 3,767.40 euros .

- **Gastos de equipos y software:** Para llevar a cabo el estudio empírico, se adquirió un portátil para ambos investigadores. El cual se considera como una herramienta fundamental para llevar a cabo las tareas requeridas. El ordenador elegido para realizar estas tareas fue un MacBook Pro 14' [10] con un chip M2 Pro, 16GB de RAM y 1TB de almacenamiento. El precio del ordenador es de 3,049 euros, el ciclo de vida son 7 años y suponiendo que un año laboral son 2,080 horas:

  $$\frac{(240 + 20) horas}{2{,}080 \frac{horas}{año} * 7\ años} * 3{,}049\ euros = 54.45\ euros$$

[9]Glassdoor: https://www.glassdoor.es/Sueldos/investigador-sueldo-SRCH_KO0,12.htm

[10]MacBook Pro: https://www.apple.com/es/shop/buy-mac/macbook-pro/14-pulgadas-gris-espacial-chip-m2-pro-de-apple-con-cpu-de-12-n%C3%BAcleos-y-gpu-de-19-n%C3%BAcleos-1tb

En cuanto a software se adquirió Microsoft 365[11] para poder crear toda la documentación del experimento, el cual su precio oficial es de 7 euros al mes o 70 euros al año. Para este trabajo se eligió la opción mensual, ya que solo se usó durante dos meses, por lo tanto: $7\frac{euros}{mes}\ x\ 2\ meses = 14\ euros$ Además del office 365 se usó Inkscape, pero este programa es gratuito. El gasto total en recursos de equipos y de software es de 68.45 euros .

- **Gastos de instalaciones:** En temas de instalaciones, todas las tareas se realizaron en la Universidad San Jorge y no se tuvo que realizar ningún pago para poder usarlas por lo tanto el gasto en esta área es de 0.

	Gasto (€)
Recursos Humanos	3,767.40 €
Equipos y Software	68.45 €
Instalaciones	0 €
Total	**3,835.80€**

Tabla 5: Gasto Total del experimento

Por lo tanto, el total de todos los gastos fue de 3,835.80 euros.

8.2. Beneficios económicos al usar SPL

El Instituto de Ingeniería de Software de la Universidad Carnegie Mellon [7], ha podido ver que las SPLs ofrecen una gran cantidad de beneficios económicos. Estos beneficios han sido reconocidos y aprovechados en CSE debido a su impacto positivo en la productividad, los costes y el tiempo de comercialización.

En términos de productividad, se ha demostrado que las SPLs pueden multiplicarla hasta por 10. Esta mejora se debe a la capacidad de reutilizar componentes de software comunes en diferentes productos de la línea. Al aprovechar la reutilización, los desarrolladores pueden reducir el esfuerzo necesario para desarrollar y mantener múltiples variantes de software. Como resultado, la productividad se incrementa considerablemente, permitiendo a las organizaciones lograr más en menos tiempo.

[11] Microsoft 365: https://www.microsoft.com/es-es/microsoft-365

Otro aspecto importante es la reducción de costes. Los informes indican que las SPLs pueden reducir los costes hasta en un 60%. Esto se debe a la reutilización de componentes, lo que evita la duplicación de trabajo y recursos. Al compartir y adaptar componentes existentes en lugar de desarrollar desde cero, se ahorran costosos recursos, como el tiempo y la mano de obra necesarios para desarrollar y mantener soluciones personalizadas.

Además de la productividad y la reducción de costes, las SPLs también tienen un impacto en el tiempo de comercialización de nuevas variantes de software. Según los informes, este tiempo puede reducirse hasta en un 98%. La reutilización de componentes permite un desarrollo más rápido y eficiente, lo que acelera la introducción de nuevos productos y características al mercado. Esta ventaja temporal proporciona a las organizaciones una ventaja competitiva al responder rápidamente a las demandas y necesidades del mercado.

9. Conclusiones

En este trabajo se llevó a cabo un estudio empírico sobre las aplicaciones de las Líneas de Producto Software (SPL) en el campo de la ingeniería de software para videojuegos (GSE). El objetivo principal fue evaluar dos enfoques de desarrollo diferentes, conocidos como Clone and Own (CaO) y SPL, en términos de Corrección, Eficiencia y Satisfacción. Para ello, se seleccionó el caso de estudio del videojuego comercial Kromaia. Para el experimento se contó con la participación de un grupo de 28 participantes, tanto expertos como inexpertos en el desarrollo de videojuegos. Esta diversidad en el perfil de los participantes permitió obtener una visión más amplia y representativa de los diferentes enfoques analizados.

Los resultados obtenidos en esta investigación revelaron que los jefes desarrollados mediante el uso de SPL fueron más precisos y correctos en comparación con aquellos desarrollados mediante el enfoque CaO. Esto demuestra las ventajas de las SPL para mejorar la calidad y la exactitud en el desarrollo de videojuegos. Sin embargo, en lo que respecta a Eficiencia y Satisfacción, no se encontraron cambios significativos entre ambos enfoques. Aunque las SPLs presentan ciertas ventajas en términos de Corrección en CSE, se ve que no son igualmente eficientes o satisfactorias en el contexto de los videojuegos. Esto sugiere la necesidad de replantear el papel de las SPLs en GSE y explorar nuevas direcciones de investigación.

Es importante destacar que este estudio pone de manifiesto la relevancia de las SPLs en el campo de la generación de nuevos contenidos de videojuegos. Al permitir la creación y gestión eficiente de variantes y características específicas, las SPLs pueden contribuir a enriquecer la diversidad y la experiencia de juego. Asimismo, se plantea la posibilidad de utilizar las SPLs como una herramienta para abordar problemas fundamentales en el diseño de videojuegos, como el balance de dificultad, ya que facilitan la incorporación de ajustes y adaptaciones.

En resumen, este estudio resalta las ventajas de las SPLs en términos de Corrección en el desarrollo de videojuegos, pero también plantea la necesidad de considerar otros factores como Eficiencia y Satisfacción. Asimismo, se señala la importancia de explorar nuevas direcciones de investigación y aprovechar el potencial de las SPLs en la generación de contenidos y la resolución de problemas fundamentales en la ingeniería de software para videojuegos.

Este trabajo se recogió en un artículo científico (Anexo IV: Publicación) que fue enviado y aceptado en la 26th ACM International Systems and Software Product Line Conference (SPLC), celebrada en septiembre de 2022 en Graz, Austria. La conferencia SPLC está clasificada según el

ranking GII-GRIN-SCIE 2021[12] como clase 2 (eventos muy buenos equiparables a JCR Q3, Q4). El artículo fue presentado por mí en la conferencia.

Posteriormente, se nos invitó a presentar una extensión de este artículo para la revista Journal of Systems and Software (JSS), clasificada según el ranking JCR de 2021 en la posición 29/110 (Q2) de la categoría *computer science, software engineering*. Actualmente el artículo se encuentra en proceso de revisión (En el Anexo V: Trabajo Futuro hay más información de este trabajo).

[12] GII-GRIN-SCIE: https://scie.lcc.uma.es/

Bibliografía

[1] A. Schultheiß, P. M. Bittner, T. Kehrer y T. Thüm, «On the use of product-line variants as experimental subjects for clone-and-own research: a case study,» de *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume A*, 2020.

[2] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon y A. Egyed, «Enhancing clone-and-own with systematic reuse for developing software variants,» de *Software Engineering 2016, Fachtagung des GI-Fachbereichs Softwaretechnik, 23.-26. Februar 2016, Wien, Österreich*, 2016.

[3] E. Ghabach, M. Blay-Fornarino, F. E. Khoury y B. Baz, «Clone-and-Own Software Product Derivation Based on Developer Preferences and Cost Estimation,» Nantes, 2018.

[4] J. Rubin y M. Chechik, «A framework for managing cloned product variants,» de *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, 2013.

[5] J. M. Horcas, M. Pinto y L. Fuentes, «Empirical analysis of the tool support for software product lines,» *Softw. Syst. Model.,* vol. 22, p. 377–414, 2023.

[6] P. Clements y L. M. Northrop, Software product lines - practices and patterns, Addison-Wesley, 2002.

[7] S. E. Institute, *Software Product Lines Collection.*

[8] K. Pohl y A. Metzger, «Software Product Lines,» V. Gruhn y R. Striemer, Edits., Cham, Springer International Publishing, 2018, p. 185–201.

[9] J. Krüger, W. Fenske, T. Thüm, D. Aporius, G. Saake y T. Leich, «Apo-games: a case study for reverse engineering variability from cloned Java variants,» 2018.

[10] C. Lima, I. do Carmo Machado, E. S. de Almeida y C. von Flach G. Chavez, «Recovering the product line architecture of the apo-games,» 2018.

[11] M. Sierra, M. C. Pabón, L. Rincón, A. A. N. Newball y D. Linares, «A Comparative Analysis of Game Engines to Develop Core Assets for a Software Product Line of Mini-Games,» 2019.

[12] L. Pascarella, F. Palomba, M. D. Penta y A. Bacchelli, «How is video game development different from software development in open source?,» 2018.

[13] SlashData, *Global developer population report 2019,* 2019.

[14] U. Technologies, *Unity Real-Time Development Platform | 3D, 2D VR & AR Engine,* 2005.

[15] E. Games, *Unreal Engine: The most powerful real-time 3D creation tool,* 1998.

[16] Crytek, *CRYENGINE | The complete solution for next generation game development by Crytek,* 2002.

[17] D. Zinoviev, «Mapping DEVS Models onto UML Models,» *CoRR,* vol. abs/cs/0508128, 2005.

[18] B. Selic, «Models, Software Models and UML,» de *UML for Real - Design of Embedded Real-Time Systems*, L. Lavagno, G. Martin y B. Selic, Edits., Kluwer, 2003, p. 1–16.

[19] M. Zhu y A. I. Wang, «Model-driven Game Development: A Literature Review,» *ACM Comput. Surv.,* vol. 52, p. 123:1–123:32, 2020.

[20] J. E. Rumbaugh, I. Jacobson y G. Booch, «The unified modeling language reference manual,» 1999.

[21] G. Sebastián, R. Tesoriero y J. A. Gallud, «A domain specific language notation for a language learning activity generation tool,» *Multim. Tools Appl.,* vol. 80, p. 36275–36304, 2021.

[22] M. Mernik, J. Heering y A. M. Sloane, «When and how to develop domain-specific languages,» *ACM Comput. Surv.,* vol. 37, p. 316–344, 2005.

[23] D. Blasco, C. Cetina y O. Pastor, «A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study,» *Inf. Softw. Technol.,* vol. 119, 2020.

[24] D. Blasco, J. Font, M. Zamorano y C. Cetina, «An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering,» *Journal of Systems and Software,* vol. 171, p. 110804, 2021.

[25] Á. Domingo, J. Echeverría, O. Pastor y C. Cetina, «Evaluating the Benefits of Model-Driven Development - Empirical Evaluation Paper,» 2020.

[26] Á. Domingo, J. Echeverría, Ó. Pastor y C. Cetina, «Comparing UML-based and DSL-based Modeling from Subjective and Objective Perspectives,» 2021.

[27] F. M. B. Boaventura y V. T. Sarinho, «MEnDiGa: A Minimal Engine for Digital Games,» *Int. J. Comput. Games Technol.,* vol. 2017, p. 9626710:1–9626710:13, 2017.

[28] V. T. Sarinho, A. L. Apolinário y E. S. Almeida, «A Feature-Based Environment for Digital Games,» Berlin, 2012.

[29] D. Castro y C. Werner, «Rebuilding games at runtime,» 2021.

[30] J. Debbiche, O. Lignell, J. Krüger y T. Berger, «Migrating Java-based Apo-Games into a composition-based software product line,» 2019.

[31] L. M. Nascimento, E. S. de Almeida y S. R. de Lemos Meira, «A Case Study in Software Product Lines - The Case of the Mobile Game Domain,» 2008.

[32] C. Lima, W. K. G. Assunção, J. Martinez, I. do Carmo Machado, C. von Flach G. Chavez y W. D. F. Mendonça, «Towards an Automated Product Line Architecture Recovery: The Apo-Games Case Study,» 2018.

[33] C. Lima, C. Chavez y E. S. de Almeida, «Investigating the Recovery of Product Line Architectures: An Approach Proposal,» Cham, 2017.

[34] L. Marchezan, W. K. G. Assunção, G. K. Michelon, E. Herac y A. Egyed, «Code smell analysis in cloned Java variants: the apo-games case study,» de *SPLC '22: 26th ACM International Systems and Software Product Line Conference, Graz, Austria, September 12 - 16, 2022, Volume A*, 2022.

[35] J. Åkesson, S. Nilsson, J. Krüger y T. Berger, «Migrating the Android Apo-Games into an annotation-based software product line,» de *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*, 2019.

[36] R. A. F. Moreira, W. K. G. Assunção, J. Martinez y E. Figueiredo, «Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases,» *Empirical Software Engineering,* vol. 27, 2022.

[37] J. Martinez, X. Tërnava y T. Ziadi, «Software Product Line Extraction from Variability-Rich Systems: The Robocode Case Study,» New York, NY, USA, 2018.

[38] E. Albassam y H. Gomaa, «Applying software product lines to multiplatform video games,» 2013.

[39] K. Constantino, J. A. Pereira, J. Padilha, P. Vasconcelos y E. Figueiredo, «An Empirical Study of Two Software Product Line Tools,» Setubal, 2016.

[40] D. Dermeval, T. Tenório, I. I. Bittencourt, A. Silva, S. Isotani y M. Ribeiro, «Ontology-based feature modeling: An empirical study in changing scenarios,» *Expert Systems with Applications,* vol. 42, pp. 4950-4964, 2015.

[41] J. Gonzalez-Huerta, E. Insfran, S. Abrahão y G. Scanniello, «Validating a model-driven software architecture evaluation and improvement method: A family of experiments,» *Information and Software Technology,* vol. 57, pp. 405-429, 2015.

[42] J. Krüger, L. Nell, W. Fenske, G. Saake y T. Leich, «Finding Lost Features in Cloned Systems,» New York, NY, USA, 2017.

[43] T. Kehrer, T. Thüm, A. Schultheiß y P. M. Bittner, «Bridging the Gap between Clone-and-Own and Software Product Lines,» Virtual, 2021.

[44] I. Reinhartz-Berger y A. Sturm, «Comprehensibility of UML-based software product line specifications - A controlled experiment,» *Empir. Softw. Eng.,* vol. 19, p. 678–713, 2014.

[45] R. Bonifácio, P. Borba, C. Ferraz y P. R. G. Accioly, «Empirical assessment of two approaches for specifying software product line use case scenarios,» *Softw. Syst. Model.,* vol. 16, p. 97–123, 2017.

[46] L. P. Tizzei, M. O. Dias, C. M. F. Rubira, A. Garcia y J. Lee, «Components meet aspects: Assessing design stability of a software product line,» *Inf. Softw. Technol.,* vol. 53, p. 121–136, 2011.

[47] A. Schlie, S. Schulze y I. Schaefer, «Recovering variability information from source code of clone-and-own software systems,» de *VaMoS '20: 14th International Working Conference on Variability Modelling of Software-Intensive Systems, Magdeburg Germany, February 5-7, 2020*, 2020.

[48] S. Adam y K. Schmid, «Effective Requirements Elicitation in Product Line Application Engineering - An Experiment,» de *Requirements Engineering: Foundation for Software Quality - 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings*, 2013.

[49] E. Figueiredo, N. Cacho, C. Sant'Anna, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. C. Ferrari, S. S. Khan, F. C. Filho y F. Dantas, «Evolving software product lines with aspects: an empirical study on design stability,» de *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, 2008.

[50] J. Echeverría, F. Pérez, J. I. Panach y C. Cetina, «An empirical study of performance using Clone & Own and Software Product Lines in an industrial context,» *Inf. Softw. Technol.,* vol. 130, p. 106444, 2021.

[51] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.

[52] A. Schultheiß, P. M. Bittner, S. El-Sharkawy, T. Thüm y T. Kehrer, «Simulating the Evolution of Clone-and-Own Projects with VEVOS,» de *EASE 2022: The International Conference on Evaluation and Assessment in Software Engineering 2022, Gothenburg, Sweden, June 13 - 15, 2022*, 2022.

[53] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell y A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.

[54] V. R. Basili y H. Dieter Rombach, «The TAME Project: Towards Improvement-Oriented Software Environments,» *IEEE Transactions on Software Engineering,* 1988.

[55] D. L. Moody, «The method evaluation model: a theoretical model for validating information systems design methods,» *ECIS 2003 proceedings,* p. 79, 2003.

[56] S. Vegas, C. Apa y N. Juristo, «Crossover designs in software engineering experiments: Benefits and perils,» *IEEE Transactions on Software Engineering,* vol. 42, p. 120–135, 2015.

[57] B. T. West, K. B. Welch y A. T. Galecki, Linear mixed models: a practical guide using statistical software, Chapman and Hall/CRC, 2014.

[58] E. I. Karac, B. Turhan y N. Juristo, «A Controlled Experiment with Novice Developers on the Impact of Task Description Granularity on Software Quality in Test-Driven Development,» *IEEE Transactions on Software Engineering,* 2019.

[59] J. I. Panach, S. España, Ó. Dieste, Ó. Pastor y N. Juristo, «In search of evidence for model-driven development claims: An experiment on quality, effort, productivity and satisfaction,» *Information and Software Technology,* 2015.

[60] F. D. Davis, «Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology,» *MIS Q.,* vol. 13, p. 319–340, September 1989.

[61] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam y J. Rosenberg, «Preliminary guidelines for empirical research in software engineering,» *IEEE Transactions on Software Engineering,* vol. 28, pp. 721-734, August 2002.

[62] M. Hendrikx, S. Meijer, J. Van Der Velden y A. Iosup, «Procedural Content Generation for Games: A Survey,» *ACM Trans. Multimedia Comput. Commun. Appl.,* vol. 9, February 2013.

[63] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis y J. Togelius, «Deep learning for procedural content generation,» *Neural Computing and Applications,* vol. 33, p. 19–37, October 2020.

[64] J. Togelius, G. N. Yannakakis, K. O. Stanley y C. Browne, «Search-Based Procedural Content Generation: A Taxonomy and Survey,» *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 3, pp. 172-186, 2011.

[65] A. J. Summerville, S. Snodgrass, M. J. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen y J. Togelius, «Procedural Content Generation via Machine Learning (PCGML),» *IEEE Transactions on Games,* vol. 10, pp. 257-270, 2018.

[66] N. A. Barriga, «A Short Introduction to Procedural Content Generation Algorithms for Videogames,» *International Journal on Artificial Intelligence Tools,* vol. 28, p. 1930001, 2019.

Tabla de Figuras

Tabla de Tablas

Anexos

Anexo I: Propuesta del Proyecto

Nombre alumno:	Jose Ignacio Trasobares Ibor
Titulación:	Ingenieria Informatica
Curso académico:	2022-2023

1. TÍTULO DEL PROYECTO

Evaluación de los beneficios de las líneas de productos de software en la ingeniería de software de videojuegos

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Las líneas de productos de Software (SPL, por sus siglas en inglés) son una forma de reutilizar software de forma sistemática utilizando el concepto de características. Las SPL han sido aplicadas en diferentes áreas, en todas ellas han demostrado tener beneficios en el coste, en eficiencia y en calidad.

La ingeniería del software para videojuegos (GSE, por sus siglas en inglés) es una de las áreas del software que más ha crecido en los últimos años. En ella se han intentado aplicar distintas técnicas de la ingeniería de software clásica. Las SPLs se han aplicado en pequeños proyectos o casos sin saber realmente si tenían algún beneficio en ella.

3. OBJETIVOS DEL PROYECTO

- Conocer el estado del arte de las SPL y de GSE.
- Diseñar y realizar una evaluación para estudiar la utilización de las SPL en GSE.
- Analizar los resultados de la evaluación para comprender el impacto que pueden tener las SPL en GSE.

4. METODOLOGÍA

La planificación se establecerá al inicio del proyecto con el tutor.

5. PLANIFICACIÓN DE TAREAS

La planificación se establecerá al inicio del proyecto con el tutor.

Anexo II: Actas de Reuniones

REUNIÓN:	**1**

Fecha:	04/03/2022		
Hora comienzo:	10:40	**Hora finalización:**	11:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega y Carlos Cetina		
Notas:	Habiendo revisado el estado actual del estado del arte en en GSE y SPL. Se decide realizar un estudio empírico que evalué si las SPLs son beneficiosas en GSE. Para ello se establece la primera tarea que es crear el estado del arte del trabajo, profundizando en ambos temas.		

REUNIÓN:	**2**

Fecha:	10/03/2022		
Hora comienzo:	15:30	**Hora finalización:**	17:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega, África Domingo y Carlos Cetina		
Notas:	Se comenta la metodología que se va a usar en este trabajo, se elige el modelo del experimento y el videojuego que se va a usar como caso de estudio. También habrá que validar el experimento en un comité de ética para poder realizarlo en la universidad.		

REUNIÓN:	**3**

Fecha:	14/03/2022		
Hora comienzo:	10:00	**Hora finalización:**	11:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega, África Domingo, Jorge Chueca, Javier Verón		
Notas:	Se convocaron a Jorge Chueca y Javier Verón, siendo desarrolladores del videojuego Kromaia, para que explicasen como era el DSL que usaban y ayudaron dándonos indicaciones para empezar a preparar los ejercicios.		

REUNIÓN:	**4**

Fecha:	16/03/2022		
Hora comienzo:	17:00	**Hora finalización:**	17:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega y Carlos Cetina		
Notas:	Se realizo esta reunión, a causa de haber problemas al poder crear la especificación de la variabilidad. Se decidió hacer varios modelos simples.		

REUNIÓN:	**5**

Fecha:	22/03/2022		
Hora comienzo:	11:00	**Hora finalización:**	12:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares y Lorena Arcega		
Notas:	Se hace revisión de todos los avances de la preparación de los experimentos y se comentan cambios para mejorar las figuras de ejemplo y solución.		

REUNIÓN:	**6**

Fecha:	29/03/2022		
Hora comienzo:	15:00	**Hora finalización:**	19:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega y África Domingo		
Notas:	Se realiza la prueba piloto y se comenta los resultados obtenidos de la prueba. Al ver los resultados comentamos que modificaciones hay que realizar a los ejercicios.		

REUNIÓN:	**7**

Fecha:	04/04/2022		
Hora comienzo:	11:00	**Hora finalización:**	12:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega y África Domingo		
Notas:	Se revisan los ejercicios con los cambios comentados en la reunión anterior, se repasa el procedimiento del experimento y se reparten los roles para el experimento.		

REUNIÓN:	**8**

Fecha:	12/04/2022		
Hora comienzo:	10:00	**Hora finalización:**	10:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se revisa la rúbrica creada, antes de empezar a corregir los ejercicios. Se ve que la rúbrica es correcta y se da la aprobación para empezar corregir.		

REUNIÓN:	**9**

Fecha:	19/04/2022		
Hora comienzo:	10:00	**Hora finalización:**	12:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se revisa el artículo que se va a mandar a SPLC y se indica los últimos cambios antes de enviarlo.		

REUNIÓN:	**10**

Fecha:	21/02/2023		
Hora comienzo:	18:00	**Hora finalización:**	19:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se comenta la información que se va a extraer del articulo y se decide ampliar el estado del arte.		

REUNIÓN:	**11**

Fecha:	26/04/2023		
Hora comienzo:	11:00	**Hora finalización:**	12:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se ven los avances de la memoria y se decide continuar por la introducción, metodología, resumen y abstract.		

REUNIÓN:	**12**

Fecha:	05/05/2023		
Hora comienzo:	10:30	**Hora finalización:**	11:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se ven los avances de la memoria y se decide empezar las secciones de desarrollo.		

REUNIÓN:	**13**

Fecha:	29/05/2023		
Hora comienzo:	10:00	**Hora finalización:**	11:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se revisan los avances de la memoria y se decide empezar el resto de las secciones.		

REUNIÓN:	**14**

Fecha:	06/06/2023		
Hora comienzo:	12:00	**Hora finalización:**	13:00
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se revisa todo el proyecto y se indican todos los cambios que hay que realizar		

REUNIÓN:	**15**

Fecha:	23/06/2023		
Hora comienzo:	11:30	**Hora finalización:**	12:30
Lugar:	Universidad San Jorge		
Elabora acta:	Jose Ignacio Trasobares Ibor		
Convocados:	Jose Ignacio Trasobares, Lorena Arcega		
Notas:	Se revisa todo el proyecto y se indican todos los cambios que hay que realizar		

Anexo III: Documentación del Experimento

Además de este documento, también se incluye un enlace a toda la documentación del experimento para que esté disponible toda la información del experimento, en el enlace están los siguientes elementos:

- **Comité de ética:** Como se comentó en la sección 5, para poder realizarse este experimento en la universidad, se tuvo que someter a un comité de ética el experimento para poder asegurarse. Esta información está en la carpeta "EthicsCommitteeApproval".
- **Objetos del experimento:** Estos son la documentación que se le dieron a los participantes, tanto los documentos de información que se les dieron de ayuda como los ejercicios que tuvieron que realizar. La información está en la carpeta "EXPERIMENTAL_OBJECTS".
- **Resultados:** Esta documentación son los resultados que se obtuvieron del experimento, en ella se incluye las correcciones de los ejercicios realizados por los participantes, como los comentarios realizados en el *Focus Group.* La información está en la carpeta "RESULTS".
- **Análisis estadístico:** Estos archivos son los resultados del análisis estadístico de los datos obtenidos en el experimento. La información está en la carpeta "STATISTICAL_ANALYSIS".

La documentación esta disponible en este enlace: http://svit.usj.es/SPLvsCAO

Figura 9: Lista de carpetas de la documentación del experimento

Anexo IV: Publicación

Evaluating the Benefits of Software Product Lines in Game Software Engineering

Se trata del trabajo presentado a lo largo de este proyecto, esta publicado y presentado en la 26th ACM International Systems and Software Product Line Conference (SPLC 2022). SPLC es el principal foro de encuentro para profesionales, investigadores y académicos dedicados a las líneas de productos de software. En esta conferencia se exponen y debaten las ideas más novedosas e innovadoras en el ámbito de las líneas de productos de software.

Jose Ignacio Trasobares, África Domingo, Lorena Arcega, and Carlos Cetina. 2022. Evaluating the benefits of software product lines in game software engineering. In Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A (SPLC '22), Vol. A. Association for Computing Machinery, New York, NY, USA, 120–130. https://doi.org/10.1145/3546932.3546998

Anexo V: Trabajo Futuro

El siguiente paso de este trabajo, fue hacer una expansión de este mismo trabajo el cual se centraría en hacer un estudio empírico, pero en este caso sería en la parte código en vez de modelos. Ese trabajo se mandó a la revista JSS en un *special issue* para extender trabajos presentados en SPLC. El resumen de este trabajo es el siguiente:

El desarrollo de videojuegos es una de las industrias de mayor crecimiento en el mundo, casi la mitad de los desarrolladores del mundo se dedican al sector de los videojuegos. Los videojuegos presentan características que diferencian su desarrollo del desarrollo de software clásico. Por ejemplo, los motores de videojuegos permiten a los desarrolladores trabajar siguiendo un paradigma de Desarrollo Basado en Modelos (MDD) o un paradigma clásico de Desarrollo Basado en Código (CDD); sin embargo, los desarrolladores de videojuegos perciben más dificultades que otros desarrolladores que no son de videojuegos a la hora de reutilizar código. Las líneas de productos de software (SPL) han demostrado su eficacia a la hora de fomentar la reutilización sistemática para desarrollar software a menor coste, en menos tiempo y con mayor calidad. Hay investigaciones recientes que proponen aplicar las SPL en el ámbito de los videojuegos. En este trabajo, se comparan enfoques para la reutilización que han funcionado en la Ingeniería de Software Clásica (CSE), aplicados a la Ingeniería de Software de Juegos (GSE) utilizando Kromaia, un videojuego comercial, como caso de estudio. Específicamente, se presenta una evaluación empírica comparando dos enfoques de desarrollo, Clone and Own (CaO) y SPLs en términos de Corrección, Eficiencia y Satisfacción cuando los sujetos desarrollan elementos de un videojuego comercial bajo dos paradigmas de desarrollo (MDD y CDD). Los resultados indican que los elementos desarrollados utilizando el SPL son más correctos (más del 23%) que los desarrollados con CaO bajo ambos paradigmas. Bajo CDD, hay mejoras significativas en Eficiencia (51%) y Satisfacción (13%) a favor del SPL, pero no hay mejoras cuando se trabaja bajo MDD. Para los desarrolladores de juegos, el impacto de utilizar SPL o CaO es mayor cuando se trabaja bajo CDD que cuando se trabaja bajo MDD. Los hallazgos también sugieren que los SPL en la GSE pueden desempeñar un papel diferente al que han desempeñado durante décadas en la CSE. En concreto, los SPL pueden ser relevantes a la hora de generar nuevos contenidos de videojuegos o de equilibrar la dificultad de estos.