Publicado en octubre 2024 por
Pedro Parrilla Bascón

http://www.lsi.us.es/~trinidad
pparrilla10@gmail.com|pedparbas@alum.us.es

Pon aquí cuestiones acerca del copyright

Yo, D. Pedro Parrilla Bascón con NIF número 77927669N,

**DECLARO**

mi autoría del trabajo que se presenta en la memoria de este trabajo fin de grado que tiene por título:

*Rogue Cards,*
*No subtitle*

Lo cual firmo,

Fdo. D. Pedro Parrilla Bascón
en la Universidad de Sevilla
07/10/2024

*Tu dedicatoria aquí*

No olvides añadir una nota de agradecimiento a quienes hayan contribuido emocionalmente al proyecto fin de Grado.

Se plantea un proyecto de Trabajo de Fin de Grado basado en el desarrollo de un videojuego de libre elección, siguiendo una pautas técnicas proporcionadas por el tutor. Entre estas directrices se encuentran indicaciones como las siguientes:

- Deberá ser un desarrollo en dos dimensiones: Los desarrollos en 2D se recomiendan a desarrolladores principiantes ya que en primera instancia es considerablemente más fácil de desarrollar.

- Se usará un motor e desarrollo conocido por la industria del videojuego: Facilita el desarrollo proporcionando herramientas y sistemas genéricos ya creados y listos para usar sobre los que se pueden desarrollar más sistemas personalizados para nuestro juego.

- El videojuego desarrollado deberá ser por turnos: De nuevo se busca el método más sencillo para presentar un videojuego ya que eliminan componentes de sincronización de sistemas que puedan requerir de arquitecturas y sistemas más complejos que puedan repercutir negativamente en el tiempo de desarrollo, reduciendo la cantidad de funcionalidades o sistemas de juegos desarrollados en la entrega.

- El alcance del proyecto deberá ajustarse a unas 300 horas de trabajo: Es el tiempo estimado que deberían durar los Trabajos de Fin de Grado. La entrega deberá contener el mayor número de funcionalidades que se hayan podido desarrollar, así como la elaboración de una Memoria sobre el desarrollo del proyecto.

Todo esto se ha condensado en la elaboración de un videojuego de combate por turnos con cartas, desarrollado en

. El juego consiste en peleas contra enemigos que controla el propio ordenador donde el jugador podrá manejar las acciones en forma de cartas que realiza un personaje elegido al inicio de la partida. Las cartas conformarán distintos "mazos.[en]tre los que el jugador podrá elegir antes de empezar la partida.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 1.1 El mundo del Videojuego

El videojuego es a día de hoy una de las herramientas más populares de entretenimiento o arte (como algunas personas defienden) que existe en el mundo. Su uso se expande tanto mundialmente como generacionalmente siendo los jóvenes los usuarios más comunes, aunque existen gran cantidad de videojuegos dirigidos a un público más adulto, cuestionando moralmente al jugador, proponiendo puzles complejos, definir estrategias, historias profundas y emocionantes, sobre la amistad, la pérdida. Mientras otros buscan una diversión fútil, para "pasar el rato"mientras esperas al autobús, o no apetece hacer nada.

## 1.2 Historia de los videojuegos

El Videojuego es un ámbito relativamente joven si se compara con otros tipos de entretenimiento que han existido a lo largo de la historia contando con apenas 75 años desde la creación de las primeras máquinas que ofrecían una experiencia interactiva sobre una pantalla que apenas podían considerarse como videojuegos.

A lo largo de los años y sobretodo a medida que la tecnología avanzaba, los videojuegos se han vuelto uno de los métodos más comunes de entretenimiento primeramente gracias a las consolas, posteriormente ordenadores, llegando hasta hoy día que tenemos acceso a infinidad de los mismos desde nuestros dispositivos móviles.

Junto al desarrollo de la tecnología los videojuegos también se han desarrollado considerablemente. Desde un simple partido de tenis con dos barras a los lados que mueves verticalmente y una bola que poder golpear como el Pong hasta la simulación del comportamiento de planetas y mundos imaginarios, pasando por la generación de contenido procedural (considerado como infinito muchas veces) y poder disfrutarlo en línea junto a otros jugadores.

## 1.3 Desarrollo de Videojuegos

El desarrollo de videojuegos se ha ido popularizando y modernizando también con los años. Los desarrolladores han ido formando empresas o grupos llamados "estudios de videojuegos"que les daba un nombre bajo los que publicar sus creaciones. Produciendo tanto los videojuegos finales como herramientas de desarrollo de videojuegos.

A lo largo de lo años se han ido desarrollando tecnologías y el desarrollo de motores de videojuegos como Unity, Unreal Engine, CryEngine e inlcuso soluciones open source como Godot o LÖVE entre otros. Esto ha facilitado una entrada más sencilla a muchos desarrolladores que quisieran explorar este mundo del videojuego.

Finalmente, mencionar que el desarrollo de videojuegos ha podido mutar hasta lo que conocemos hoy en día como juegos independientes o Ïndie"que son producciones hechas por desarrolladoras pequeñas o incluso un solo individuo. Promoviendo así esta modalidad de la informática y haciéndola más accesible. Como ejemplos podemos mencionar: The Game Kitchen, desarrolladora española con sede en Sevilla que ha desarrollado Blasphemous. O juegos mundialmente conocidos como Undertale de Toby Fox, Stardew Valley de Eric ÇoncernedApe"Barone.

## 1.4 Estado del arte

El mundo del videojuego hoy en día es un ecosistema vivo y en constante evolución. La industria ha crecido enormemente, generando miles de millones de dólares en ingresos. Podemos destacar varias tendencias:

- Diversificación de plataformas: Los juegos no solo se juegan en consolas y PCs, sino también en dispositivos móviles, ampliando el acceso y audiencia de los mismos.

- Desarrollo de juegos Indie's: Los desarrolladores independientes han ganado terreno, creando experiencias innovadoras que a menudo desafían las normas de la industria. Destacando "Hollow Knight.º Çeleste.$^{en}$tre otros, muy populares y aclamados tanto por su creatividad como jugabilidad.

- E-Sports: La competencia profesional en videojuegos se ha profesionalizado, con grandes torneos y ligas que atraen a millones de expectadores y producen grandes beneficios para las empresas detrás de los mismos. Destacan juegos como "League of Legends", "Valorant.º "Fortnitez Çlash Royale".

- Realidad Virtual y Aumentada: La tecnología VR y AR está en auge, ofreciendo experiencias inmersivas. Juegos como "Beat Saber.º "Pokemon GO"muestran el potencial de estas tecnologías.

- Modelos de Negocio: Si bien se pueden asemejar al modelo de cualquier producto de entretenimiento más convencional como libros, de un solo pago, el videojuego permite nuevos modelos como las microtransacciones a cambio de beneficios o cos-

méticos dentro del juego o las subscripciones como los servicios de Netflix u otras plataformas de subscripción de pago, donde el contenido se va actualizando periódicamente. Vease las plataformas como Xbox Game Pass o PlayStation Plus. Aparte de los juegos que se actualizan en contenido como "World of Warcraftz sus expansiones donde añaden mundos nuevos y misiones a completar. Grandes empresas dedican parte de sus esfuerzos dentro de este ámbito de los videojuegos. Colosos como Sony, Microsoft, Steam, Epic Games o incluso Amazon agrupan a gran cantidad de estudios que desarrollan y publican sus juegos en sus plataformas.

- Narrativa y Experiencia: Los videojuegos se están reconociendo cada vez más como una forma de arte, con narrativas profundas, experiencias emocionales y gráficos realistas como pueden ser "The Last Of Us.º "Baldur's Gate 3çon sus detallados mundos postapocalíptico y fantástico respectivamente.

- Inclusividad y diversidad: Hay un esfuerzo creciente por hacer que los juegos sean más inclusivos, con personajes diversos y narrativas que abordan temas sociales relevantes hoy en día.

- Educación: Algunas desarrolladoras han visto el potencial educativo de los videojuegos no solo para niños sino también para el desarrollo, formación y práctica de otros sectores como la salud. Encontramos ejemplos como "Minecraft Education.º "VR Surgery Simulator".

Esta imagen proporciona una perspectiva muy buena para el mundo del videojuego en los próximos años frente a la oleada de despidos y estudios cerrados que ha habido durante los últimos años .y . Este efecto se le atribuye a la estabilización del mercado tras la inflación que sufrió a partir de la pandemia del 2020 donde se popularizó masivamente el uso de los videojuegos como método de entretenimiento haciendo que se crearan estudios masivamente y se produjeran videojuegos por encima de las espectativas.

Sin embargo, el mundo del videojuego está lejos de desaparecer, cada año obteniendo más beneficios y generando productos de cada vez mayor calidad como el aclamado por la crítica "Baldur's Gate 3"que ha superado todas las espectativas tanto en ventas como en calidad a la hora de producción. Incluso poniendo en jaque a muchos estudios que publican sus juegos en estados que podrían ser casi calificados como ïnjugables".

## 1.5 Premisa del proyecto

El proyecto se basa en la premisa de un desarrollo de videojuego completo hecho en Unity. Para lo cual se ha propuesto un videojuego de cartas por turnos contra la máquina. Estas características han sido escogidas bajo previa aprobación del tutor del proyecto, coincidiendo con los criterios proporcionados para la correcta realización del trabajo.

Se toma el desarrollo de videojuego como reto personal y profesional del autor. Es un ámbito complejo que requiere de amplios conocimientos y habilidades tanto técnicas, creativas como sociales o humanas. Adquiriendo experiencia en áreas como la programación o la gestión de proyectos. Además, como se afronta desde la perspectiva de un novato dentro del ámbito del videojuego, se aprenderá desde cero.

Se espera un desarrollo de habilidades transferibles como la resolución de problemas y gestión de tiempo, altamente valoradas en la industria tecnológica entre otras. Así como el inicio de contrucción de un portafolio de proyectos personales que puedan ayudar en el futuro a respaldar mi experiencia y habilidades.

Por último destacar la satisfacción personal de realizar un proyecto de esta índole, siendo los videojuegos una pasión para mí y viendo realizado en forma de experiencia jugable una idea propia.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Aquí va un breve resumen del capítulo.*

## 2.1 Aprendizajes del proyeto

La idea del proyecto es el aprendizaje y familiarización en el desarrollo de videojuegos desde cero, utilizando un motor gráfico moderno como Unity, así como realizar una planificación y gestión de un proyecto informático de forma profesional, demostrando así los conocimientos adquiridos durante el Grado de Ingeniería de Software.

## 2.2 Motivación

El auge de los juegos de cartas y roguelikes ha generado un gran interés en mecánicas que combinen estrategia y rejugabilidad. Este es un género de juegos que si bien no es el más popular, obtiene gran cantidad de fama y buena crítica. Esto se debe a su alta rejugabilidad, ya que cada partida es única gracias a la generación aleatoria de niveles y cartas, lo que mantiene el interés de los jugadores haciendo que cada partida sea distinta haciendo que el jugador tenga que adaptarse a la aleatoriedad. Además, ofrecen una profunda estrategia en la construcción de mazos, lo que atrae a quienes disfrutan del pensamiento crítico. Su accesibilidad permite que nuevos jugadores aprendan las mecánicas de forma gradual, mientras que su estética atractiva y narrativas interesantes ayudan a sumergir a los jugadores en el juego. Las comunidades activas y el contenido adicional que se crea alrededor de estos, junto con el éxito crítico, han fomentado un ambiente de apoyo y conexión entre jugadores. Por último, la diversidad de estilos de juego asegura que haya algo para todos los gustos, desde desafíos serios hasta experiencias más ligeras, consolidando su lugar en el panorama actual de los videojuegos.

Sin embargo, muchos juegos del género se sienten repetitivos o carecen de profundidad en la construcción de mazos o diversas mecánicas que potencialmente puedan mejorarlo. El desafío es crear una experiencia que ofrezca variedad en cada partida, manteniendo la emoción y la sorpresa, al tiempo que se fomente la toma de decisiones estratégicas.

### 2.2.1 Estudio de mercado



Como referencias para este proyecto se han usado juegos que han tenido una gran aceptación entre la comunidad de jugadores. Estos son "Slay the Spire", Rogue Book.º "Balatro":

**Slay the Spire**

Concepto Básico: Slay the Spire es un juego roguelike de cartas en el que los jugadores eligen un personaje y deben ascender por una torre donde deberán elegir el camino a seguir, cada camino siendo distinto, enfrentándose a enemigos y jefes en cada nivel. La clave es construir un mazo de cartas a partir de las que van obteniendo a lo largo del camino.

Destaca por su profundidad estratégica, permitiendo crear mazos que tengan mucha sinergia, es decir, los efectos de unas cartas combinan con el resto haciendo que la experiencia se vuelva adictiva a medida que se completa el mazo. Esto unido a que cada uno de los personajes jugables tiene su propio mazo de cartas hace que la experiencia sea prácticamente única todas las partidas.

Elementos Incluidos:

**Múltiples Clases:** Cada personaje tiene su propio conjunto de cartas y habilidades únicas.

**Generación Procedural:** Los mapas, encuentros y recompensas se generan aleatoriamente, garantizando que cada partida sea diferente.

**Eventos y Encuentros:** Durante su ascenso, los jugadores pueden encontrar eventos aleatorios que ofrecen decisiones que afectan su progreso.

**Relación entre Cartas:** La mecánica de construcción de mazos permite sinergias entre cartas, lo que enriquece la estrategia.

**Categorización de cartas:** Las cartas tienen tres categorías: Ataques, Habilidades y Poderes. Tanto las cartas como los enemigos pueden reaccionar a los tipos de cartas jugadas.

Como puntos a tener en cuenta para nuestro proyecto, tendremos: - Variedad de efectos de cartas. Y reacciones a cartas jugadas. Teniendo cuidado para no abrumar al jugador con mecánicas complejas que entorpezcan la entrada al juego. - La rejugabilidad, con cada partidad siendo completamente aleatoria. - Equilibrio entre dificultad y progresión, para mantener a los jugadores desafiados pero no frustrados. - Narrativa sutil mediante eventos y textos que no cortan la que añade una capa de inmersion sin interrumpir el flujo del juego.

**Roguebook**

Concepto Básico: Roguebook combina elementos de roguelike y construcción de mazos, permitiendo a los jugadores explorar un mundo en un formato de libro abierto. Los jugadores controlan a dos personajes simultáneamente, combinando sus cartas para enfrentar enemigos.

Destaca por su exploración "libre"dentro de los límites de un mapa, y su combinación de cartas, ya que cada personaje jugable tiene su propio mazo y se juega con dos personajes a la vez, mezclando sus mazos a la hora de robar cartas". Esto hace que cada partida sea potencialmente un problema distinto y que tengas que adaptarte a las fortalezas de cada personaje e intentar aprovechar al máximo su simbiósis. Esto unido a la toma de decisiones estratégicas sobre los recursos de "tinta"que descubren partes del libro (el mapa) lo vuelven en un juego que plantea problemas de difícil solución.

Elementos Incluidos:

**Sistema de Combinación de cartas:** Los jugadores pueden mezclar cartas de dos héroes, creando sinergias y estrategias personalizadas.

**Exploración del Mapa:** Los mapas son dinámicos y permiten a los jugadores explorar diferentes caminos, lo que añade una capa de estrategia.

**Estilo Visual:** Presenta una estética de ilustración encantadora que da vida al mundo del juego.

**Eventos Aleatorios:** Incluye eventos y desafíos inesperados que ofrecen recompensas o complicaciones adicionales.

Podemos entonces tener en cuenta: - Estética atractiva y vibrante que destaca en el género. - Combinaciones de personajes con mazos distintos. - Cartas con efectos simples y menos sinergias que Slay the Spire. - La crítica comenta que la dificultad puede ser inconsistente lo que puede llevar a momentos de frustracion.

**Balatro**

Concepto Básico: Balatro es un juego de cartas roguelike donde los jugadores juegan al poker buscando superar una serie de puntuaciones mínimas para superar varios niveles de apuestas a través de combinar manos de póker con los "joker"que aportan modificaciones sobre las mismas.

Aporta un enfoque divertido y accesible al género gracias a su fácil comprensión de mecánicas y simpleza de interfaz que lo vuelven adictivo. Además cuenta con un gran sistema de desbloqueo de cartas, haciendo que el jugador cumpla algún tipo de reto que podría mermarle a corto plazo pero abrirle más posibilidades de juego en otras partidas. Creando así un sistema que se retroalimenta haciendo que el jugador tenga que debatir sobre la mejor estrategia.

Elementos Incluidos:

**Mecánica de Apuestas:** Los jugadores pueden arriesgar cartas en situaciones específicas para obtener beneficios o potenciadores, lo que añade una dinámica de riesgo y recompensa.

**Temática Ligera:** El juego tiene un enfoque ligero, las consecuencias de perder no son muy relevantes, se enfoca en hacer pasar el tiempo de una forma entretenida y adictiva.

**Estrategia de construcción de mazos:** Similar a otros roguelikes de cartas, los jugadores deben construir y optimizar su mazo adaptándose a la situación concreta y aleatoria de la partida.

**Interfax simple:** El diseño de la interfaz es amigable para nuevos jugadores, facilitando la comprensión de las mecánicas del juego.

**Familiaridad:** Al estar basarse en el póker, los nuevos usuarios ya están familiarizados con las cartas.

Tras analizarlo podemos concluir entonces: - Estética simple, más fácil de entender para los jugadores. - Mecánicas adictivas, el jugador puede arriesgar cartas para obtener beneficios, añadiendo un elemento de riesgo-recompensa. - En término de cartas, aunque sea más fácil de entender, las mismas cartas todas las partidas puede ser repetitivo, afectando a su rejugabilidad a largo plazo. - Muchas veces el juego puede sentirse injusto por mecánicas como la aleatoriedad en "niveles"(enemigos) que pueden explotar ciertas debilidades de los mazos. Esto repercute negativamente si no está bien equilibrado, haciendo que la estrategia y habilidad del jugador se sienta insignificante.

## 2.3 Propuesta

Se propone un Juego de cartas por turnos donde el jugador maneje un personaje elegido antes de comenzar una partida. Este a su vez elegirá un mazo de cartas con el que jugar y progresar. El jugador avanzará por unos niveles donde encontrará enemigos que querrán terminar su partida antes de lo esperado si no los derrota mediante las acciones definidas en las cartas.

### 2.3.1 Sistema de estadísticas

Todos los personajes tanto jugables como enemigos tienen unas estadísticas que influirán en el desarrollo de las acciones. Estas estadísticas están basadas en el sistema conocido de Dragones y Mazmorras 5ª edición. Gran parte del proyecto toma ciertos aspectos de este conocido juego de mesa y las digitaliza y adapta.

Estas reglas hacen que las estadísticas tomen ciertos valores comprendidos entre 1 y 20 y estas afecten al resto de acciones realizables durante el juego según defina si son necesarias. ¿Cómo las modifican? Usando un valor conocido como Modificador (Mod) que es el resultado de restarle 10 al valor de la estadística (Stat), dividirlo entre dos y aplicarle una función suelo.

$$Mod = \lfloor (Stat - 10) \div 2 \rfloor$$

**Estadísticas**

spacing

Fuerza(Strength, STR): Representa la fuerza del personaje e influye tanto en el ataque como en la defensa con cartas de fuerza.

Destreza(Dexterity, DEX): Representa la agilidad del personaje e influye tanto en el ataque como la defensa con cartas de destreza.

Constitución(Constitution, CON): Representa la fortaleza y resistencia del personaje influendo tanto en la Salud que tendrá el personaje así como en las cartas que utilicen esta estadística y la cantidad de acciones por turno que tendrá.



Inteligencia(Intelligence, INT): Representa la destreza mental en combate así como el conocimiento del personaje influyendo en las carta que usen las características

así como en las cartas obtenidas al inicio de ronda y las posibilidades de robo de carta.

Carisma(Charisma, CHA): Representa la astucia y carácter del personaje influye en las cartas que usen la estadística así como en las interacciones sociales que el jugador pueda ir encontrando a lo largo de la partida.

Los personajes también tienen unos atributos que pueden influir a lo largo de la partida como: - Nivel: medirá el progreso e influirá en otras estadísticas. - Puntos de vida base: serán los puntos de vida a nivel 1 e influirán en el cálculo de la vida cuando se suba de nivel. - Inmunidades: Lista de tipos de daño que no afectan al personaje. - Resistencias: Lista de tipos de daño que afectan la mitad al personaje. - Debilidade: Lista de tipos de daño que afectarán el doble al personaje.

Los enemigos contarán con 3 atributos extra: - Daño base: El daño que se usará para calcular la fuerza de un ataque de este enemigo. - Tipo de daño de ataque: El tipo de daño producido por el ataque. - Habilidad de ataque: Referencia a la estadística con la que este enmigo calcula la fuerza de ataque, sumándola al daño base.

Los héroes o personajes jugables también contarán con la estamina que determinará el total de los costes de las acciones que el jugador podrá realizar por turno.

**Sistema de dados**

RogueCards usa un sistema de dados similar al de Dragones y Mazmorras. Estos se usarán para determinar el poder de los efectos que se dan lugar durante la partida. Existen 7 tipos de dados, cada uno de ellos devuelve valores entre [1, X] siendo X el valor mostrado en el nombre del dado: - d4 - d6 - d8 - d10 - d12 - d20 - d100

Los dados que se usen pueden implementar una mecánica de explosión o crítico. Esta mecánica se basa en que cuando se obtiene el valor máximo del dado, si el efecto es explosivo, el dado volverá a tirarse, sumándose a la tirada anterior y pudiendo explotar de nuevo infinitamente.

**Sistema de cartas**

Las cartas están divididas en 3 secciones: Ataques: aplicarán daño sobre los enemigos, Habilidades: aplicarán efectos como curación y escudo al héroe, Poderes: Modificarán las estadísticas del héroe. Las cartas contienen cierta información que será usada para calcular las acciones que estas producen durante la partida: - Nombre. - Coste de estamina: Representa el gasto de estamina que le supone al jugador usar la carta.

- Dado: Representa el dado que se usará para calcular el resultado de jugar la carta. - Explosivo: Determina si el efecto es explosivo. - Estadística: Determina la estadística del personaje que influirá en el cálculo del efecto. - Los ataques también incluyen el tipo de daño realizado.

Las cartas estarán alojadas en mazos que se podrán elegir de entre 3 predefinidos al principio de la partida, simulando estilos de juegos típicos de los juegos de rol, como son el Pícaro(Rogue), Guerrero(Berserker) o Mago(Mage). Cada uno con sus habilidades y armas específicas (cartas).

## 2.4 Listado de objetivos

Tras este análisis podemos elaborar una lista de objetivos que puedan dirigir el desarrollo de forma que podamos evitar los errores que cometen los juegos mejor valorados dentro del género y aprovechar sus fortalezas ya consolidadas dentro de los jugadores típicos de este género de Roguelike de cartas por turnos":

**Objetivo Final. Desarrollo de un Producto Mínimo Viable** Crear un prototipo de videojuego con mecánicas parecidas, y similares a las de los videojuegos de cartas estudiados previamente.

**Objetivo 1. Conceptualización y diseño inicial** Plantear unas directrices sobre cómo debe funcionar el juego, qué mecánicas hay que desarrollar y cómo. A su vez establecer un estilo visual y temática que tenga sentido dentro del proyecto.

**Objetivo x. Preparación y familiarización con Unity** Creación de un proyecto vacío sobre el que poder iniciar el desarrollo así como conocer sus sistemas y cómo funciona.

**Objetivo x. Definición de sistemas de juego** Desarrollo de las estadísticas o los sistemas base que formarán la base del juego.

**Objetivo x. Movimiento de cartas** Creación de un sistema que nos permita mover las cartas por la escena, así como tener una "mano"de carta.

**Objetivo x. Creación de un sistema de cartas redu** Creación de cartas, sus datos y componentes visuales para poder tener una representación prácticamente jugable con la que poder probar el resto de sistemas.

**Objetivo x. Desarrollo del sistema de mazos** Sistema sobre el que se desarrollan los robosƶ "descartes"de las cartas, así como la elección del mazo jugable.

**Objetivo x. Desarrollo del sistema de personajes** Creación de los sistemas de datos y representación de personajes tanto aliados como enemigos.

**Objetivo x. Desarrollo del sistema de combate** Elaboración de un sistema mediante el que poder jugar las cartas y aplicar los efectos de las mismas.

**Objetivo x. Desarrollo del sistema de turnos** Poder terminar el turno del jugador y que la máquina controle a los enemigos. A su vez cambiando de ronda y habilitando de nuevo al jugador.

**Objetivo x. Desarrollo de condiciones de Victoria y Derrota** Hacer que el jugador pueda ganar o perder la partida.

**Objetivo x. Desarrollo de sistema de progresión de partida** Crear un progreso dentro de una partida.

**Objetivo x. Desarrollo del sistema de progresión de personaje** Crear el progreso de personaje para que pueda evolucionar a medida que progresa la partida.

**Objetivo x. Desarrollo sistema de progresión de mazo** Desarrollar un sistema mediante el que añadir cartas nuevas al mazo que esté jugando

**Objetivo x. Creación del ejecutable del juego** Obtener un producto final que puedas ser jugado y probado por un grupo de jugadores y obtener feedback para mejorar el juego hasta llegar a un punto donde pueda ser lanzado.

Todos estos objetivos definen a su vez una sucesión de puntos de control que están ordenados teniendo en cuenta las necesidades del proyecto en cada momento. Por ejemplo para validar el "Desarrollo de condiciones de Victoria y Derrota", el sistema de combate o el de turnos debe estar desarrollado, si no podemos tener un sistema que no se puede probar cómoda y correctamente evitando así la posible deuda técnica generada al no tener sistemas probados.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 3.1 Estructura organizacional del proyecto

El proyecto se realiza en solitario siendo Pedro Parilla Bascón el autor en su plenitud del proyecto. Le supervisa como tutor del proyecto D. Pablo Trinidad.

## 3.2 Metodología de desarrollo

### 3.2.1 Feature Driven Development (FDD)

Para el desarrollo del proyecto vamos a basarnos en la metodología conocida como Feature Driven Development (FDD). Esta es una metodología ágil de desarrollo de software centrada en la construcción de funcionalidades específicas. En lugar de seguir un enfoque tradicional basado en la documentación o el diseño completo antes de la implementación, FDD se centra en entregar características concretas en ciclos cortos de desarrollo. Esto permite adaptarse rápidamente a los cambios y proporciona un producto que puede ser evaluado y mejorado continuamente.

**Características de FDD**

Funcionalidades FDD se basa en identificar y desarrollar características concretas del sistema. Cada funcionalidad debe ser valiosa para el usuario y está alineada con los requisitos del negocio.

rdeso Modelo Global Se crea un modelo conceptual que describe la estructura del sistema y sus principales componentes.

ista de Funcionalidad Se define una lista detallada de las características que el sistema debe tener.

ón por Funcionalidad Las funcionalidades se dividen en tareas manejables y se planifican en ciclos cortos.

ño por Funcionalidad Cada funcionalidad se diseña y se desarrollan las especificaciones necesarias.

ón por Funcionalidad Se implementa y prueba cada funcionalidad de manera individual.

eraciones Cortas favorece ciclos de desarrollo cortos (generalmente de 1 a 2 semanas), lo que permite una entrega continua de funcionalidades y permite adaptarse a cambios rápidamente.



Roles claros Se definen roles específicos en el equipo, como el Chief Architect (arquitecto principal), que se encarga del modelo global, y los Feature Teams, que son responsa-

bles de implementar funcionalidades específicas.

mentación ligera La documentación se centra en las características y sus diseños, en lugar de extensos documentos de requisitos o especificaciones. Esto facilita la comprensión y la comunicación.

visión y Reportes Al final de cada iteración, se busca feedback de los clientes/usuarios, lo que permite ajustar y mejorar el producto según las necesidades y expectativas de los mismos.

### 3.2.2 Adaptación

A partir de las características previamente expuestas vamos a seleccionar y curar las que más nos puedan ayudar para el correcto desarrollo del proyecto, ya que FDD se orienta e implementa en su totalidad cuando existe un equipo de desarrollo y unos clientes más al uso.

**Selección de procesos**

funcionalidades Identificar las características clave que se desea implementar, como "sistema de movimiento de cartas", "turnos", etc...

osición en Tareas Dividir cada funcionalidad en tareas más pequeñas y manejables. Por ejemplo, para el "sistema de movimiento de cartas", definiríamos tareas como ïmplementar lógica de arrastre", "definir interacciones con el entorno".

ón de iteraciones Establecer un calendario para desarrollar y entregar cada funcionalidad en iteraciones cortas de una o dos semanas. Al final de cada iteración se deberá tener una funcionalidad completamente operativa y probada.

Revisión Tras la iteración, probar que el sistema funciona correctamente

Integración Asegurar que la nueva funcionalidad es compatible con el resto de sistemas previamente implementados

Documentación Se generará la documentación necesaria para la correcta entrega del TFG. Intentando no dar explicaciones sobre implementaciones completas a bajo nivel en lo que a código respecta.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 4.1 Resumen temporal del proyecto

| Resumen del proyecto | |
|---|---|
| **Fecha de inicio** | 06/07/2024 |
| **Fecha de fin** | 09/10/2024 |
| **Periodicidad de las revisiones** | 1 previa a la entrega |
| **Carga de trabajo semanal** | 24,5 horas |
| **Horas totales previstas** | 300 horas |
| **Horas finales** | 300 horas |

Cuadro 4.1: Tabla resumen de tiempos y planificación

## 4.2 Planificación inicial

Aquí un desglose de las iteraciones, comienzo y fin de cada una:

| Resumen de iteraciones | |
|---|---|
| **Iteración 1** | 06/07/24 a 21/07/24 |
| **Iteración 2** | 22/07/24 a 07/08/24 |
| **Iteración 3** | 08/08/24 a 26/08/24 |
| **Iteración 4** | 27/08/24 a 10/09/24 |
| **Iteración 5** | 11/09/24 a 01/10/24 |

Cuadro 4.2: Planificación temporal de iteraciones

El criterio seguido para la separación de los periodos de iteración es el siguiente:

Vacaciones: Debido a que trabajo a la vez que desarrollo el proyecto, se va a tomar como referencia las vacaciones aprobadas y definir toda la planificación en base a esas dos semanas que se ven representadas en la tercera iteración. Estas representarán la mayor carga de trabajo debido al aumento de tiempo disponible para dedicarle al proyecto.

Empleo: En el caso de coincidir con una semana de aumento de la carga de trabajo, como medida preventiva, se ha decidido alargar las iteraciones hasta un máximo de 3 días naturales

manas naturales: Se intentará terminar cada iteración con cada semana natural, puesto que los días no laborables se podrá dedicar más tiempo a resolver problemas, probar e integrar los progresos que hayan habido durante la iteración.

### 4.2.1 Desglose de iteraciones según planificación inicial

**Iteración 1**

**ESTE CAPÍTULO DEBE ESCRIBIRSE AL COMIENZO DEL PROYECTO**

## 4.3 Informe de tiempos del proyecto

Lo mismo que el anterior pero con datos reales. Ver Tabla **??**.

| Resumen de iteraciones | |
|---|---|
| **Iteración 1** | 10/10/14 a 21/10/14 |
| **Iteración 2** | 21/10/14 a 15/11/14 |
| **...** | dd/mm/aa a dd/mm/aa |

Cuadro 4.3: Planificación temporal de iteraciones

Justificar los retrasos de forma detallada aquí para cada una de las iteraciones. Explicar las razones.

problemas durante el drag de las cartas ' displacement implementacion arquitectura de soa

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 5.1 Resumen de costes del proyecto

| Resumen del proyecto | |
|---|---|
| **Costes de personal** | 5.045 € |
| Sueldo neto | 2.030 € |
| Impuestos | 1.000 € |
| Costes sociales | 2.015 € |
| **Costes materiales** | 560 € |
| **Costes indirectos** | 450 € |
| **TOTAL** | 8.000 € |

Cuadro 5.1: Tabla resumen de costes

## 5.2 Costes de personal

Ya hablaremos de esto

## 5.3 Costes materiales

Y de esto también. Ver Sección **??**.

## 5.4 Costes indirectos

Y esto es una fiesta

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 6.1 Lista de características

Aplicar aquí la primera iteración de Feature Driven Development.

## 6.2 Diseño arquitectónico

Descripción de los sistemas de producción, preproducción y pruebas.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 7.1 Características a desarrollar

1. Funcionalidad A. Ver Tabla **??**.

2. Funcionalidad B.

| Análisis de valor aportado 0001 | |
|---|---|
| **Propuesta** | Trabajo que pretende analizarse y justificarse |
| **Valor** | Qué valor aporta al proyecto o al usuario final. |
| **Coste** | Qué costes en términos de esfuerzo, adquisiciones y limitaciones tiene la propuesta |
| **Opciones** | Qué otras opciones se tienen que aporten un valor similar? ¿Es realmente un valor relevante para el proyecto/cliente |
| **Riesgos** | Qué riesgos pueden surgir a la hora de desarrollar esta propuesta. |
| **Deuda técnica** | Posibles deudas técnicas que se asumen con el desarrollo de esta propuesta. |

Cuadro 7.1: Análisis de valor aportado 0001

## 7.2 Diseño

Aquí una discusión de cómo va a afectar todo al diseño

Debe insertarse un diagrama UML de diseño con los cambios y hacer referencia en el texto así Fig. **??**.

Un memorando técnico por cada decisión de diseño.

## 7.3 Implementación



Un memorando técnico por cada decisión de implementación y refactorización que afecte al diseño.

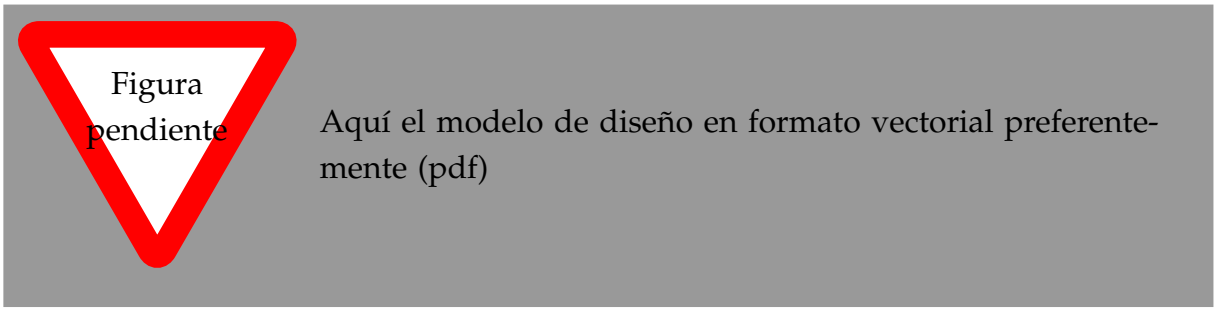
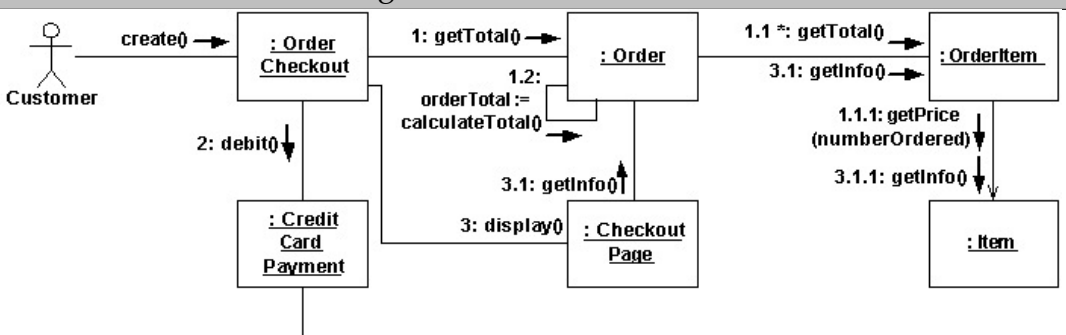



Figura 7.1: Diagrama UML de diseño para la iteración 1

| Memorando técnico 0001 | |
|---|---|
| **Asunto** | ¿Cuál es el problema? |
| **Resumen** | ¿Cuál es la solución propuesta? |
| **Factores causantes** | Descripción pormenorizada del problema |
| **Solución** | Descripción pormenorizada de la solución propuesta |
| **Motivación** | ¿Por qué propone esta solución? |
| **Cuestiones abiertas** | Factores a tener en cuenta en la solución cuya dimensión se reconoce. |
| **Alternativas** | Otras soluciones consideradas y la razón por la que se excluyeron. |

Cuadro 7.2: Memorando técnico 0001

| Identificador | Descripción de la acción de alto nivel | | |
|---|---|---|---|
| 0001 | Prueba | | |
| Métodos de alto nivel | | | |
| [return_type] method_name1 (param1:type1, ...) | | | |
| Pasos (Usar Pseudocódigo o similar) | | | |
| 1. Paso 1. | | | |
| 2. Paso 2. | | | |
| Métodos de bajo nivel necesarios | | | |
| **Paso** | **Clase** | **Método** | **Mem. Técn.** / **IU** |
| 1 | ClassName | [return_type] me | 001 / CL |
| Diagrama de Colaboración | | | |



| Identificador | Descripción de la acción de alto nivel |
|---|---|
| alvotermar02 | Grubber |
| Métodos de alto nivel | |
| [return_type] grubber (param1:type1, ...) | |
| Pasos (Usar Pseudocódigo o similar) | |
| 1. Lanzar 2 dados | |
| 2. Compara resultado de los dados con kicking del open-side | |
| 2.1. Si valor dados es menor o igual a kicking, avanza 10m | |
| 3.1. Si no hay defensa y el golpeo es exitoso, el pateador retiene la posesión del balón | |
| 3.2. Si hay defensa y el golpe es exitoso, el atacante tira un dado y suma su valor al de speed y strength y el defensor lanza 2 dados y lo suma al valor de speed y strength de su jugador, el vencedor será aquel que tenga más puntos, si es igual, la posesión es del defensor | |
| 4.1. Si no es exitoso y hay defensa el balón pasa a posesión del defensor | |
| 4.2. Si no es exitoso y no hay defensa de lanza un line-out | |
| Métodos de bajo nivel necesarios | |

| **Paso** | **Clase** | **Método** | **Mem. Técn.** | **IU** |
|---|---|---|---|---|
| 1 | Dice | [Integer] throwDice () | 001 | SI |
| 2 | ClassName | [Int] compareKickingToDice (kicking:Integer, dice: Integer) | 001 | SI |
| 2.1 | ClassName | [Integer] setLine (line:Integer) | 001 | SI |
| 4.2 | ClassName | [Integer] lineOut () | 001 | SI |

## 7.4 Pruebas

Descripción de las pruebas realizadas al software

## 7.5 Despliegue

Breve resumen de cómo se han desplegado los cambios en el sistema de producción.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 8.1 Sección libre

Estructurar en función del proyecto.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?*

## 9.1 Informe post-mortem

Qué es un informe post-mortem

### 9.1.1 Lo que ha ido bien

- Argumento a favor 1.
- Argumento a favor 2.
- Argumento a favor 3.

### 9.1.2 Lo que ha ido mal

- Argumento en contra 1.
- Argumento en contra 2.
- Argumento en contra 3.

### 9.1.3 Discusión

En función de lo anterior, qué cambiaría si empezara hoy el proyecto de nuevo.

## 9.2 Trabajos futuros

Enumera los puntos abiertos y que no se han resuelto. Indica si darían lugar a otro proyecto y de qué forma se podría acotar.

*The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.*

*Ernest Hemingway (1899–1961),*
*Novelist*

*This is an example of an abstract. Multiple lines are supported. Several paragraphs. It jumps to the next page. Blau blau blau. I am introducing more text to reach the third line*

## A.1 Software Product Lines

- Objective of a Product Line (PL) (mass production and customisation) [**?** ]
- The focus in software derives in Software Product Lines (SPLs).
- Variability management: variability models
- When and how are used VMs: FMs are described in FODA report as a key element in SPL since they represent the variability and commonality of the different products in a SPL.

## A.2 Feature Models

To Abductive Section in 2.1

As the number of products to be built by a SPL may be large and the constraints among features may be complex, representing such an information in a manageable and compact manner is a must. Feature Models (FMs) represent the set of products a SPL may build in terms of product features. Some features are optional while others are mandatory. To indicate the relationships among features, they are hierarchically linked, forming a tree whose root is a feature representing the whole functionality of a product. The root feature is refined in child features, which increase the level of detail and reduce the scope of features. Recursively following this refinement process, a tree-like structure is obtained where three basic kinds of hierarchical relationships are used:

- Mandatory: a mandatory relationship affects a parent and child feature. It forces the child feature to appear in a product whenever its parent feature does.
- Optional: a child feature connected to a parent feature by means of an optional relationship may be optionally selected whenever its parent feature is.
- Set-relationships: three or more features are part of a set-relationship: a parent feature and a set of two or more child features. A set-relationship contains a cardinality that constraints the number of child features to be selected in a product whenever its parent feature is selected. If the cardinality is $[1..1]$ it is commonly remarked as an *alternative relationship* where only one child feature may be selected at the same time. If the cardinality is $[1..N]$ (where $N$ is the number of

child features), it is also known as an *or-relationship* as any combination of child features is allowed while at least one is selected.

Although FMs can represent most of the most frequent constraints, the hierarchical nature of these models might hinder the representation of some constraints. Under this circumstance, *cross-tree constraints* can be added. The most common kinds of cross-tree constraints are:

- Dependency: a feature depends on another feature if the second one must be part of a product whenever first one is selected.
- Exclusion: two features exclude themselves if both of them cannot be part of a product at the same time.

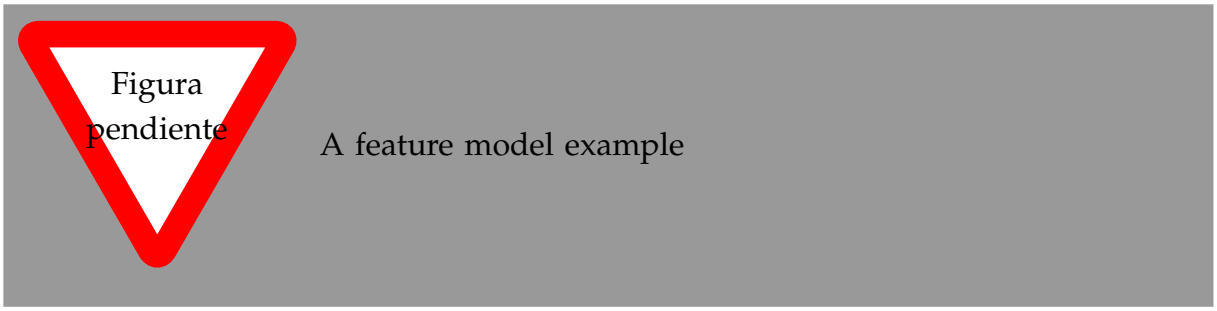



Figura A.1: An example of a Home Integration System

The example in Figure **??** describes a *Home Integration System* (HIS) SPL in terms of its features and the relationships among them. Leaning on this example we define some useful terms:

**Partial configuration** : a partial configuration is a composed by three sets of selected ($S$), removed($R$) and undecided($U$) features. A feature can only be in one of these sets and every feature in the FM ($fm$) must be in one of them, i.e. $S \cup R \cup U = fm$ and $S \cap R \cap U = \emptyset$. A partial configuration represents an intermediate state during the process of a customer selecting the feature for a custom product. For example, $S_P = \{...\}$, $R_P = \{...\}$ and $U_P = \{...\}$ define a partial configuration for the sample FM where some features are still to be decided if they are to be selecter or removed in a configuration.

**(Full) configuration** : a full configuration or simply a configuration is a partial configuration such that the set of undecided features in empty. For example, $S_F = \{...\}$ and $R_F = \{...\}$ describe a full configuration for the example FM.

**Product** : a product is a representation for a full configuration such that only the selected features are remarked. For instance, $P = \{\}$ is a product for the above full configuration. A product such as `A,B` is a valid since all the constraints within the FM are satisfied. However, A,B and C is not a valid product since D is required.

**Validation** A partial configuration is *valid* if all the relationships and constraints are satisfied given the sets of selected, removed and undecided features. So the definition applies for valid full configurations and valid products. As a conclusion we can affirm that a FM represents all the valid products in a SPL.

Objetivo: Briefly expose attributes as an important asset in feature models.

It is frequent that features are not enough to represent information that is relevant to represent a SPL variability. In this case, FMs are extended with feature attributes such as cost, versions, RAM consumption, etc. in the so-called Extended Feature Models (EFMs) [**?** ]. Besides relationships, an EFM contains constraints that affect attributes which reduce even more the set of products a FM describes. Above definitions remain when attributes are introduced into FMs.

## A.3 Automated Analysis of Feature Models

### A.3.1 Scope

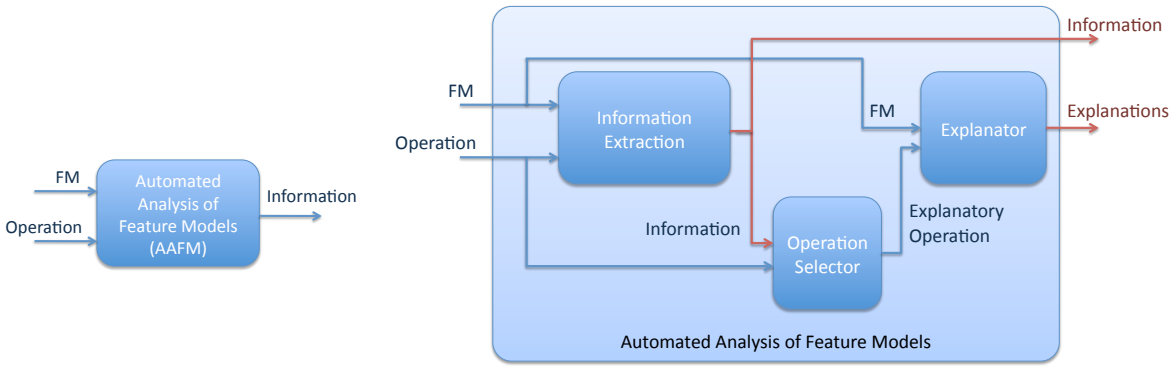
To Abductive Intro

FMs are used all along the SPL development as key models and many of the development decisions are taken relying on the information contained within them. Most of the times, relationships are complex and hinder the manual extraction of information. Manually obtaining information such as 'which is the product that costs the less?', 'does the feature model contain errors?' or 'why there exist no product containing certain features?' can be an unfeasible task. The complexity and compactness of FMs justify the need of an automated support of these operations. So the *Automated Analysis of Feature Models* (AAFM) arises as a topic of interest to deal with this problem in the SPL community.



The AAFM can be seen as a black-box process that receives a FM and an operation as inputs and obtains information (its kind depends on the analysis operation) as an

output (Fig. **??**). There are many operations that extract information from a FM such as 'counting products' operation whose result is a natural number indicating the number of customised products that can be built; or 'list of products' operation that obtains each of those products. This vision of AAFM as a black-box is valid for a subset of analysis operations that we call *information extraction operations* (IEO) that can be seen as processes to extract information from FMs. In other words, an IEO makes explicit an implicit information within a FM.




(a) The AAFM seen as a black-box process

(b) Extending the AAFM process with explanations

Figura A.2: A different view on AAFM distinguishing between information extraction and explanatory operations

Use me to explain in a larger text than 'side-text' anything that is important to a reader not familiar with the dissertation context for example.

However, there is a subset of analysis operations known as *explanatory operations* (EO) whose objective is explaining the result obtained from a IEO. Sometimes, the result is not the expected one and the analyser needs to know which are the relationships that have caused it. For example, let us suppose that the IEO 'which are the products described in a FM that cost less than $1000?' obtains no products as a result. If we were expecting to obtain at least one product, it is important to determine the relationships in the FM that are responsible of that behaviour, so an EO 'why there is no product costing less than $1000?' will shed light on the relationships that avoid obtaining any product. Obtaining no result is not the only case that claims for explanations. If we obtained only one product as a result and we were expecting to obtain at least 10 products, although an answer is obtained the result is unexpected and the discrepancy reasons have to be found. Moreover, explanatory operations are also use-

ful even when an expected result is obtained, to reinforce the certainty that the result is correct. So it can be concluded that EOs complement the information an FM analyser obtains from IEOs.

The complexity of feature modelling relies on correctly setting the relationships that describe the set of products to be built in a SPL. Relationships are the only elements responsible of the results obtained in FM analysis. So an *explanation* is a set of relationships that may have caused that result. While IEO provides for an unique response that is known for certain, an EO provides for a set of probable explanations to a result obtained from a IEO, being only one of them a valid explanation. It would be the analyser the one in charge of discriminating the correct explanation, maybe performing new analysis operations.

THIS IS A SIDE TEXT. USE TO REMARK IMPORTANT INFORMATION

Therefore, two kinds of operations are distinguished in AAFM: information extraction and explanatory operations. Explanatory operations have no sense without a paired information extraction operation and its result. To ensure that explanatory operations are always paired to an information extraction operation, we define a new black-box process of AAFM that incorporates explanations as an additional output (see Figure **??**)

1. Information extraction: the original process, which remains the same.

2. Operation selector: depending on the information extraction operation the analyser asks for and the information obtained as a result, this process provides the explanatory operation to be performed. In other words, it pairs an explanatory operation to an information extraction operation.

3. Explanatory analysis: provides a set of explanations from the FM and the explanatory operation.

The overall process can be encapsulated into a holistic black-box process which receives the FM and the information extraction operation as inputs and provides a result and explanations as outputs. It can be seen as we just add explanations as an output to the analysis process.



To realise this view on the AAFM, we need to give details on the insides of these black-boxes. Since the information extraction process is already rigourously defined in

Benavides' PhD dissertation, the purpose of this paper is defining the remaining two sub-processes. We formalise the explanatory analysis process by means of default logic and provide the criteria to implement the operation selector process.

Most Common Techniques to perform AAFM Operations.

## A.4 Dynamic Software Product Lines (DSPL)

What is a Dynamic Software Product Line (DSPL). Different points of view. What is important is the automation of reconfiguration properties relying on SPL techniques.

We focus in the application of explanations in DSPLs as an application of our results. Specifically we have worked in MAS and smart homes providing a solution for automating product reconfiguration.

## A.5 Hypothesis and Objectives

Objetivo: Justifying that explanations are a particular set of operations in AAFM that are not solvable by means of the techniques that are used up-to-date

Objetivo: Set an impacting phrase that summarises the hypothesis

**Hypothesis**

*Explanations cannot be solved by AI techniques used to solve AAFM. There should exist other AI techniques to solve explanations.*

**Objective of the dissertation**

*Defining a framework to provide solutions for explanatory analysis in FMs.*

This dissertation summarises our contribution to solve some of the objectives we set in our PhD project.

- Defining a catalog of analysis operations where explanations are applied.
- Rigorously defining these operations in terms of logics.
- Proposing solutions to these operations.
- Validating our results by means of tools and projects where they are applied.

Next chapter focuses on refining how we have contributed to deal with the above objectives.

A piece of code...

```
public Map<Cardinality,CardinalValue> detectWrongCardinals() {
        // any other implementation of Map can be used instead.
        Map<Cardinality,CardinalValue> result =
                new TreeMap<Cardinality,CardinalValue>();
        for( r : relationships) {
                if (r instanceof Set) {
                        Set set = (Set)r;
                        Cardinality card = set.getCardinality();
                        Domain dom = card.getDomain();
                        for (value: dom.getValues())
                                if (isWrongCardinal(card,value))
                                        result.put(card,value);
                }
        }
        return result;
}
```

A coolTable. Use inside a table.

Use \TableSubtitle{n,title} to add a subtitle as the header. n is the number of columns and title is the text to place. [**?** ]

| A Catalog of FM Explanatory Operations (2009 version) | | |
|---|---|---|
| **Information Extraction Operation** | **FM Explanatory Operations** | |
| | *Why? operation* | *Why not? operation* |
| Valid FM | - | invalid FM |
| Valid Configuration | valid partial conf. | invalid partial conf. |
| Valid Product | valid product | invalid product |
| Products Listing | vaild Product/Config | invalid FM/Product/Config |
| Products Counting | vaild Product/Config | invalid FM/Product/Config |
| Optimisation | vaild Product/Config | invalid FM/Product/Config |
| Core feature | core feature | core feature |
| Variant feature | variant feature | variant feature |
| Dead feature detection | - | dead feature |
| False-optional feature detection | - | false-optional feature |
| Wrong-cardinality detection | - | wrong cardinal |
| **Information Extraction Operation** | **Configuration Explanatory Operations** | |
| | *Why? operation* | *Why not? operation* |
| Valid Configuration | valid partial conf. | invalid partial conf. |

Cuadro A.1: Most frequently used explanatory operations and their corresponding information extraction operations