eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática
Computación

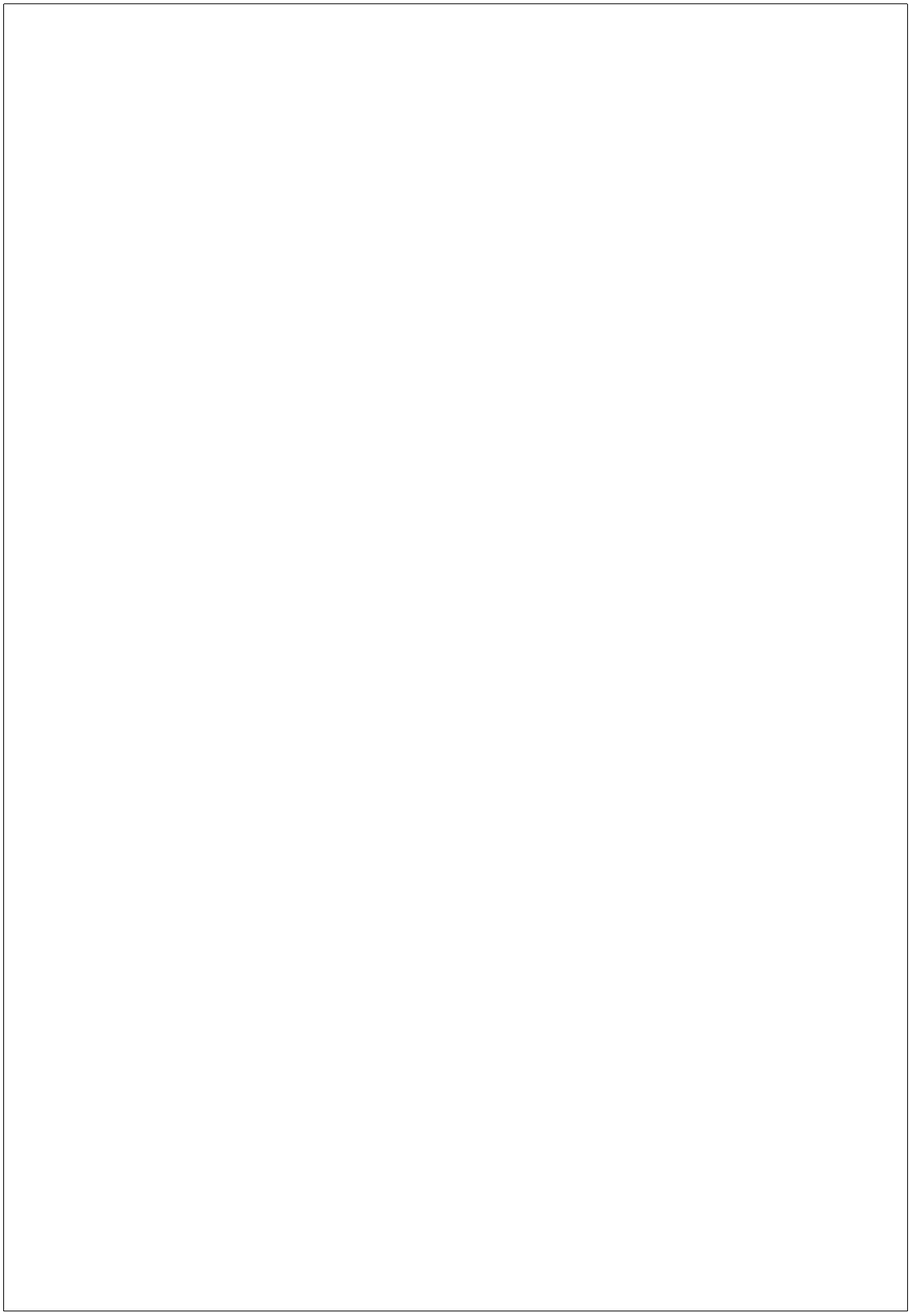
Trabajo de Fin de Grado

Transpilador de un lenguaje de modelado personalizado de sistemas de simulación dinámicos discretos a Python

Autor
Baldwin David Rodríguez Ponce

2022

BORRADOR. Nota: Márgenes del documento dibujados explícitamente.



eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática
Computación

Trabajo de Fin de Grado

Transpilador de un lenguaje de modelado personalizado de sistemas de simulación dinámicos discretos a Python

Autor
Baldwin David Rodríguez Ponce

Directora
María Begoña Losada Pereda

BORRADOR. Nota: Márgenes del documento dibujados explícitamente.

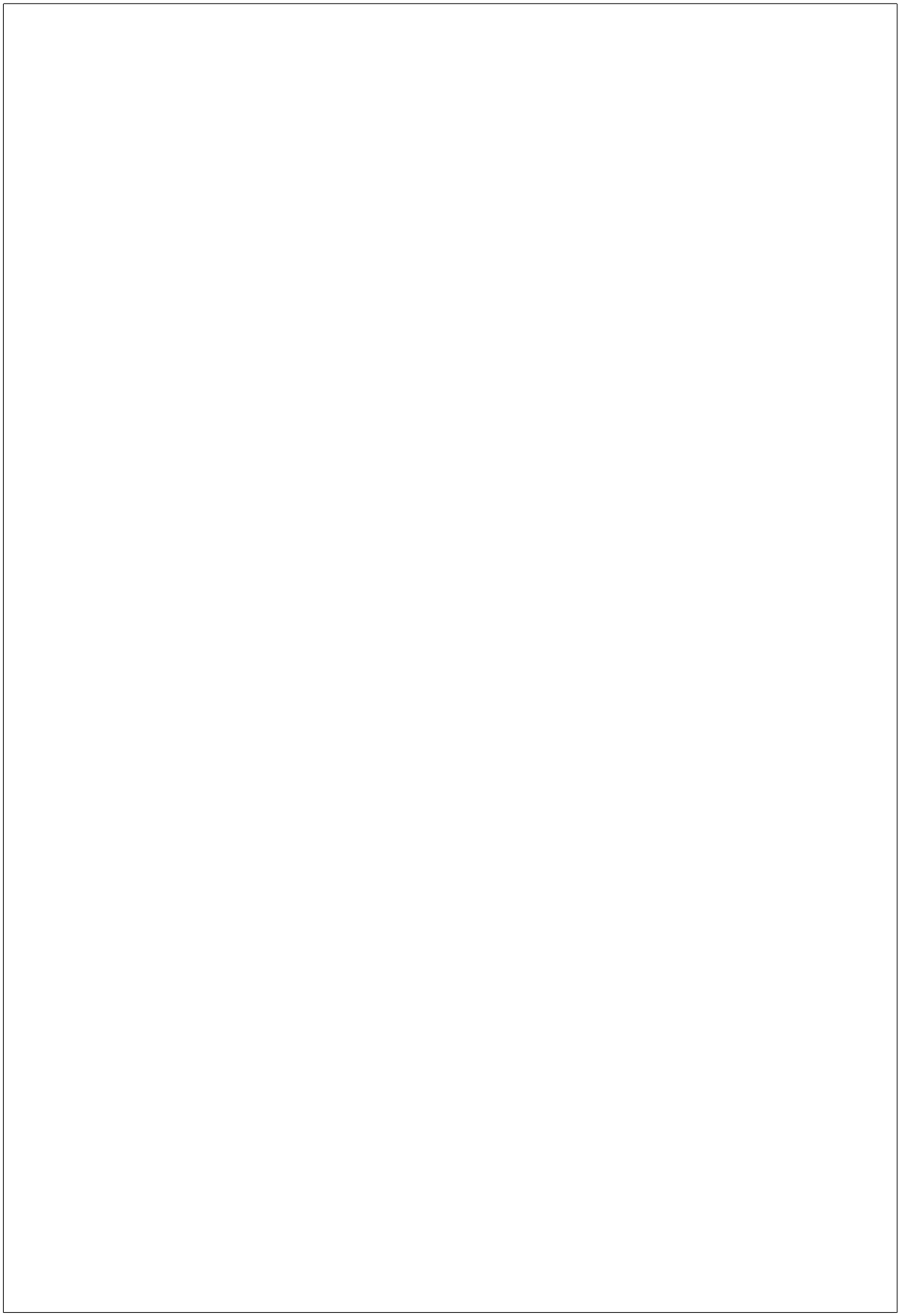
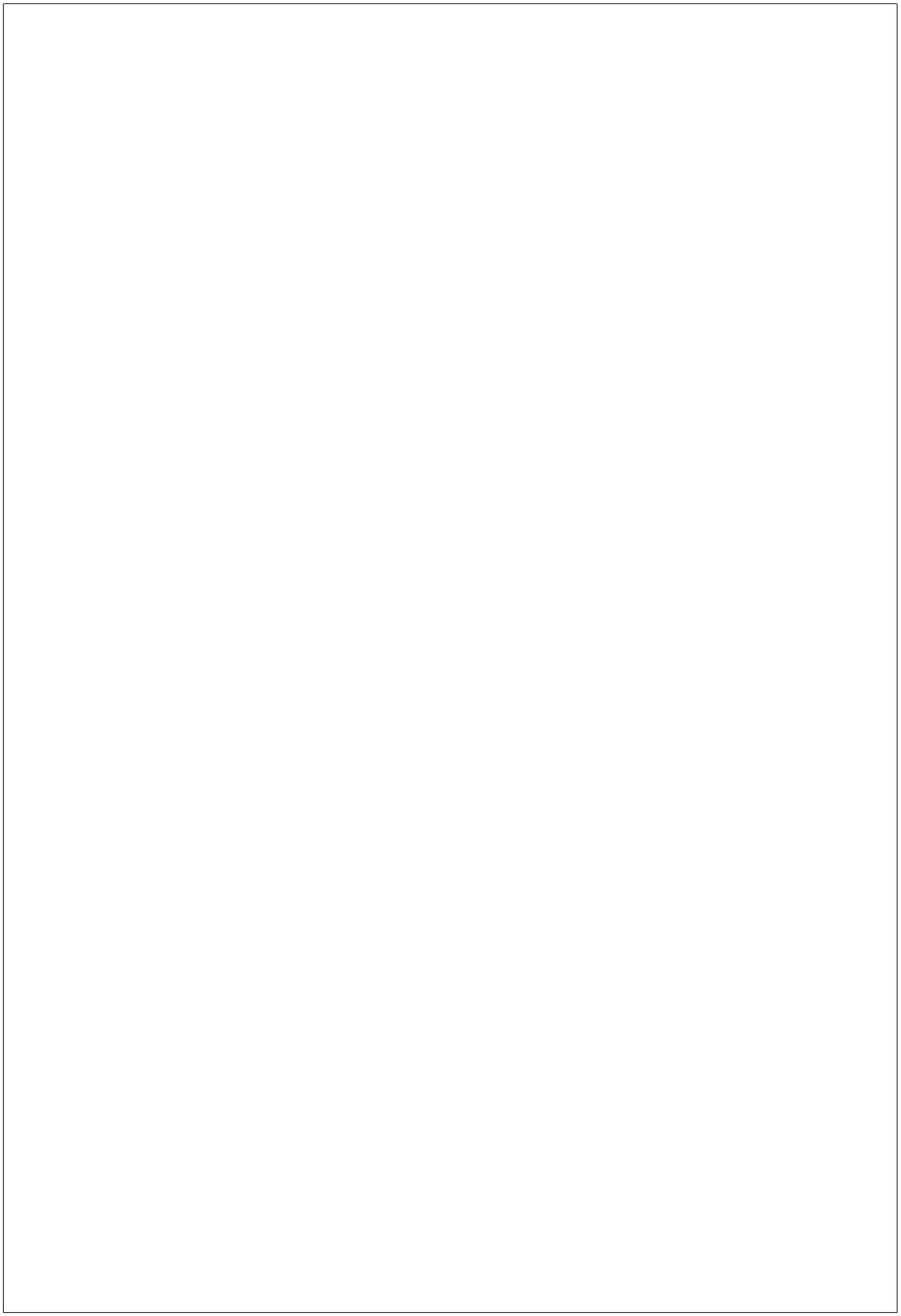
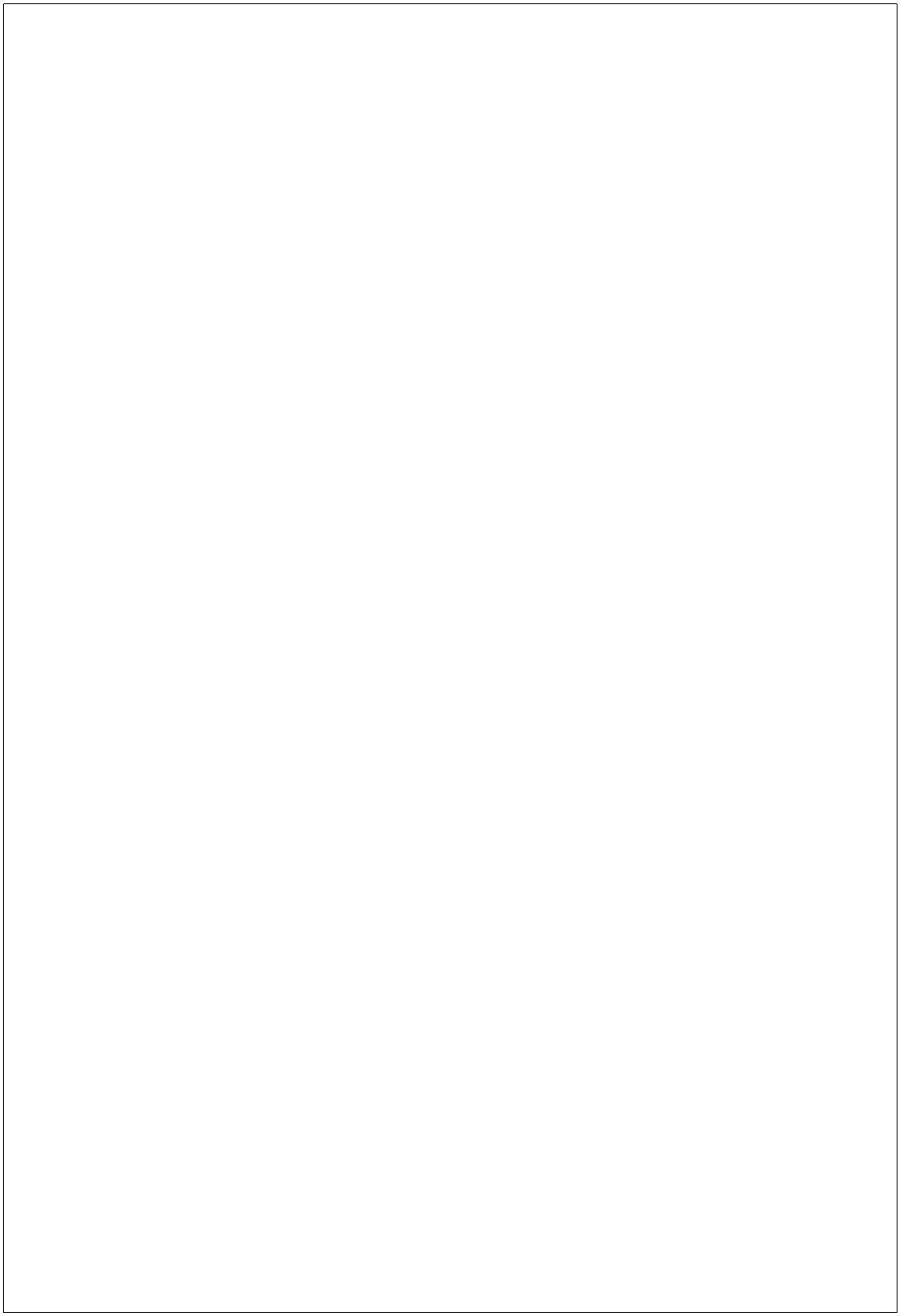


Tabla de contenidos



Índice de figuras



CAPÍTULO 1

Gestión del proyecto

La información se sacará principalmente de Project Management Institute, 2017

1.1. Alcance

El alcance de este proyecto incluye el trabajo necesario para diseñar, implementar y documentar un framework de modelado y desarrollo de simulación de sistemas dinámicos discretos basados en eventos. Dicho framework se dividirá en tres módulos principales:

- **Lenguaje**: Un lenguaje de modelado específico pensado para ser usado por usuarios que no tengan mucha experiencia en programación. Se hará uso de las herramientas de desarrollo de compiladores Flex y Bison para generar un transpilador que traduzca ficheros de este lenguaje a código Python. A través de él se plantea:
  - Permitir la rápida implementación de este tipo de modelos a través de grafos de sucesos.
  - Permitir que el programador del lenguaje se encargue sólo de realizar las implementaciones pertinentes al sistema de simulación que desee desarrollar:
    - Especificación de las variables globales, variables de entrada, contadores estadísticos y medidas de rendimiento propias del modelo.
    - Inclusión de eventos adicionales y sus acciones correspondientes.
    - Creación y eliminación de eventos en función de tiempo y condiciones lógicas.
    - Inclusión de código adicional escrito directamente en Python en caso de ser necesario.

Cambiar redacción

- **Núcleo**: Una serie de módulos que implementarán un microframework de simulación de este tipo de sistemas en específico para Python, pensado para ser usado por programadores y para acotar la traducción del nuevo lenguaje. A través de él se plantea:

Aquí te falta la posibilidad de poner las cosas variables también

- Permitir que la traducción del lenguaje incluya dentro del fichero generado las estructuras de datos, funciones y procedimientos que tienen en común todos los sistemas dinámicos discretos:
  - Generadores de datos aleatorios para distintos tipos de distribuciones.
  - Reloj y temporizador de simulación para ejecutar los eventos.
  - Estructura de datos para almacenar los sucesos según deben ocurrir en el tiempo.
  - Las respectivas implementaciones mínimas de los dos eventos que siempre formarán parte de todos los modelos: “Inicio” y “Fin”.
  - Generador de informes final que se ejecutará al finalizar la simulación y mostrará los resultados que se deseaban estudiar con ésta.

- **CLI**: Una interfaz de comandos por terminal que se usará para gestionar, configurar y ejecutar los proyectos desarrollados con este framework.

1.1.1. Objetivos

1.1.1.1. Objetivo general

Generar un lenguaje de modelado de simulación de sistemas dinámicos discretos basados en eventos junto con un transpilador que lo traduzca a Python, un microframework para acotar la traducción y un CLI para gestionar proyectos desarrollados con este producto.

1.1.1.2. Objetivos específicos

1. Diseñar un nuevo lenguaje de modelado y simulación de sistemas dinámicos discretos basados en eventos usando las herramientas de desarrollo de procesadores de lenguaje Flex y Bison.
2. Diseñar, implementar y verificar una serie de módulos y funcionalidades realizadas en Python con el fin de generar un microframework para simular estos mismos sistemas.
3. Implementar y verificar un transpilador que traduzca el lenguaje del producto a Python usando las características del objetivo anterior.
4. Diseñar, implementar y verificar una serie de operaciones accesibles desde una CLI de cara a ser usadas para la gestión, configuración y ejecución parametrizada de simulaciones realizadas con el producto.
5. Implementar distintas técnicas de análisis de salidas, experimentación y optimización de modelos dentro del núcleo del framework.
6. Desarrollar un manual de usuario que contendrá toda la documentación necesaria para hacer uso del framework.

1.1.2. Requisitos

El requisito base principal del proyecto consiste en cumplir con un tiempo de dedicación total máximo de 300 horas.

1.1.2.1. Requisitos funcionales

TERMINAR

1.1.2.2. Requisitos no-funcionales

TERMINAR

1.1.3. EDT

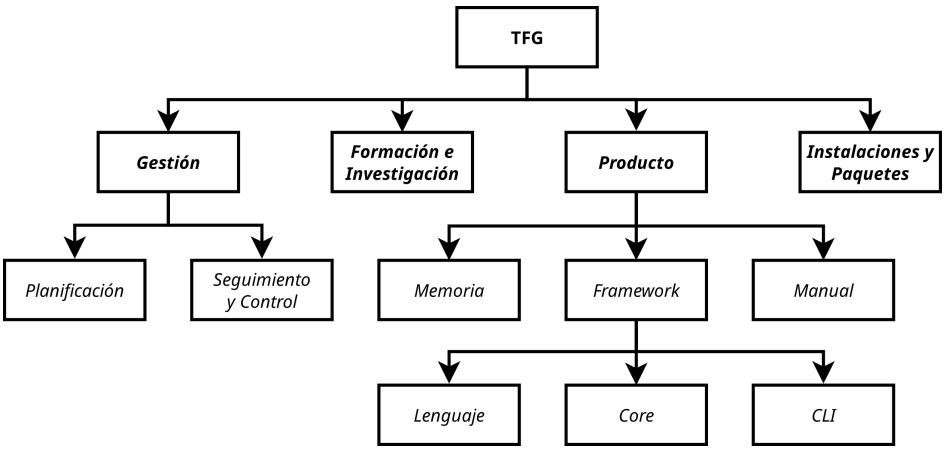


Figura 1.1: Esquema de Descomposición de Trabajo del Proyecto

1.1.3.1. Rama de Gestión

El paquete de trabajo **Gestión (Ge)** contendrá:

- **Planificación (Ge.P):** Este paquete de trabajo agrupará todas las tareas relacionadas a la realización de la planificación y preparación inicial del proyecto.
- **Seguimiento y Control (Ge.S):** Este paquete de trabajo agrupará todas las tareas necesarias para asegurar que el seguimiento del proyecto está realizando como se plantea en la planificación y, en caso de no ser así, controlar las consecuencias de las desviaciones emergentes.

1.1.3.2. Rama de Formación e Investigación

El paquete de trabajo **Formación e Investigación (FeI)** contendrá todas las tareas relacionadas con el aprendizaje de herramientas, búsqueda de referencias y recolección de información necesaria para el desarrollo del proyecto.

1.1.3.3. Rama de Producto (Pr)

El paquete de trabajo **Producto (Pr)** contendrá todas las tareas relacionadas al propio diseño, implementación y verificación de los entregables principales del proyecto. Podemos desglosarlo en tres paquetes más pequeños:

- **Memoria (Pr.Me):** Este paquete de trabajo agrupará todas las tareas necesarias para la realización de la memoria final del proyecto.
- **Framework (Pr.Fr):** Este paquete de trabajo agrupará todas las tareas relacionadas al desarrollo del framework planteado en el alcance del proyecto. Podemos separarlo en sus tres partes principales:
  - **Lenguaje (Pr.Fr.Le):** Este paquete contendrá todas las tareas relacionadas al diseño y desarrollo del nuevo lenguaje de modelado.
  - **Core (Pr.Fr.Co):** Este paquete contendrá todas las tareas relacionadas al diseño y desarrollo del propio núcleo del framework.
  - **CLI (Pr.Fr.Cli):** Este paquete contendrá todas las tareas relacionadas al diseño y desarrollo de la interfaz de comandos proporcionada para facilitar la gestión de proyectos realizados con el a generar.
- **Manual (Pr.Ma):** Este paquete de trabajo agrupará todas las tareas necesarias para la realización del manual de usuario que se generará para facilitar el uso del producto a desarrollar.

1.1.3.4. Rama de Instalación y Paquetes

El paquete de trabajo **Instalación y Paquetes (IyP)** agrupará todas las tareas relacionadas a la preparación del entorno de desarrollo y la instalación de paquetes y dependencias para la realización del producto principal del proyecto.

1.2. Metodología

A la hora de gestionar un proyecto, es recomendable definir una metodología de trabajo, seguimiento y control con el fin de asegurar el correcto cumplimiento y finalización de su lista de requisitos y objetivos. Tras investigar y plantear distintas alternativas, se ha escogido la metodología Kanban para desarrollar este proyecto.

El método Kanban como tal es útil si se desea:

- Iniciar un proyecto de manera ágil, rápida, de coste nulo y bajo riesgo.
- Analizar y mejorar el proceso de trabajo ya existente.
- Controlar múltiples grupos de tareas.
- Asegurar que el número de tareas en ejecución están dentro de un nivel aceptable.
- Cambiar a una mentalidad de desarrollo ágil.

Esta sección, por tanto, estará dedicada a explicar los conceptos e ideas necesarias de cara a justificar la razón por la cual se ha escogido este método de trabajo. Además, es necesario mencionar que esta información será resumida principalmente de Cole y Scotcher, 2015 y Stellman, 2014, siendo éstas las referencias recomendadas en caso de desear más detalles al respecto.

1.2.1. Definición

El método Kanban tiene sus orígenes en las fábricas de coches de Toyota a manos de Taiichi Ohno. Fue creado como un simple sistema de planificación y administración de trabajo e inventario de cada fase de producción. Sin embargo, fue David J. Anderson quien definió y adaptó esta metodología para el uso en ingeniería y desarrollo de software.

Es necesario mencionar que, contrario a lo que se piensa normalmente, Kanban es una metodología ágil de mejora de procesos y no un framework de administración de proyectos. Y así como muchos métodos de este tipo, se caracteriza principalmente por ser evolutivo e iterativo en el tiempo.

Asimismo, Kanban comparte la misma ideología de trabajo con el método Lean hasta tal punto que se considera que es una especialización de esta última. Aplicando los principios y valores de este proceso, Kanban se centra en eliminar los desperdicios de tiempo y recursos que el equipo tiene. Por lo tanto, este proceso de mejora pide a los equipos de desarrollo empezar con una metodología ya existente para poder perfeccionarla gradualmente en el tiempo a través de la experimentación, el cálculo de distintas métricas de rendimiendo y la confirmación de resultados positivos según dichas medidas.

Todo equipo cuenta con un sistema para la implementación de código, ya sea que se siga una metodología formal como Scrum o que se disponga de una serie de reglas no definidas o reconocidas explícitamente. Como consecuencia de esto, lo único que se necesita para empezar a utilizar Kanban es identificar el proceso de desarrollo actual para poder formalizarlo y adaptarlo a esta metodología.

Por tanto, aunque Kanban no es un sistema de gestión de proyectos, es posible hacer uso de éste para ello al tener como principal objetivo aumentar la predictabilidad del flujo de trabajo y así mejorar la planificación del proyecto como tal.

1.2.2. Principios de Kanban

Al ser una especialización del método Lean, se tiene una serie de principios directores en esta metodología. Especificamente se pueden listar los siguientes tres:

- **Empieza con lo que se hace actualmente**: Como se había mencionado anteriormente, el método Kanban pide requiere de un proceso o metodología inicial. Es a través del análisis y la comprensión de dicho proceso que Kanban permite perfeccionarlo iterativamente. Pero como consecuencia de esto, no es posible aplicar Kanban adecuadamente si se desconoce la metodología que usa el equipo de desarrollo.
- **Comprométete al cambio evolutivo e incremental**: Aunque parezca muy repetitivo,

es necesario volver a mencionar que el objetivo de Kanban es la mejora gradual del sistema. A eso se refiere el cambio evolutivo e incremental y es por eso que el método pide calcular métricas de control. Es a través de estas medidas que se llegan a conocer las partes mejores del proceso de desarrollo.

- **Respeta el proceso, los roles, las responsabilidades y títulos actuales**: Cuando el equipo de trabajo dedica tiempo a medir el rendimiento del sistema, es posible encontrar ciclos de retroalimentación que contienen información importante para la mejora evolutiva del proceso. Sin embargo, para poder aplicar dicha información es necesario conocer cómo se aplica ésta a cada rol del equipo. Por esta razón Kanban considera relevante respetar las responsabilidades y roles asociadas a cada miembro del grupo.

1.2.3. Prácticas de Kanban

El método Kanban, además, define explícitamente una serie de prácticas a llevar a cabo con el fin de aplicar correctamente la mejora de procesos que éste permite. Sin embargo, no es necesario hacer uso de todas estas en su totalidad al iniciar con el método. Kanban tiene específicamente los siguientes seis principios:

- **Definir y visualizar el flujo de trabajo**: Con el fin de familiarizarse más con el proceso de trabajo, Kanban pide hacer uso de representaciones visuales de éste a través de tablones, listas de trabajo y elementos de trabajo. La combinación de dichos componentes permite definir el flujo de trabajo en su totalidad de una manera fácilmente entendible. En Kanban, visualizar significa anotar exactamente lo que hace el equipo sin embellecer los detalles con el fin de observar el sistema en su totalidad.
- **Limitar el trabajo en progreso**: En el método Kanban los distintos trabajos a realizar en el proceso de desarrollo tienen un límite de tareas en ejecución en un determinado momento. La justificación que se le da a este principio recae en el hecho de que realizar múltiples labores a la vez reduce la eficiencia del equipo. A dicho límite se le conoce como *WiP* (de sus siglas en inglés *Work-in-Progress*) o *Trabajo en Progreso*.
- **Manejar el flujo de trabajo**: Una vez definido el flujo de trabajo, el objetivo será conseguir una rápida y suave transición entre los distintos grupos de tareas, desde la lista de trabajo a empezar hasta dar por finalizada la labor. Si se consigue esta velocidad de flujo, se dice que se está operando con la eficiencia óptima y es en este momento que se crea el máximo valor laboral en el menor tiempo posible. A medida que el equipo desarrolla, se van encontrando los cuellos de botella y ajustando los límites de trabajo.
- **Hacer explícitas las políticas de proceso**: Para poder obtener un flujo de trabajo óptimo, además, es necesario definir los objetivos que se deben cumplir para dar por terminada una serie de tareas con el fin de determinar cuándo se avanza de estado. Dichas condiciones de finalización, sin embargo, siempre irán ligadas al tipo de proceso que se está utilizando y, por tanto, deberán ser discutidas y especificadas por el equipo.
- **Implementar ciclos de retroalimentación**: Como se había mencionado anteriormen-

te, los ciclos de retroalimentación permiten identificar la información necesaria para aplicar mejoras en el proceso de desarrollo. Kanban define dichos ciclos como la combinación de limitaciones que permiten dar a conocer los puntos débiles del equipo y su correcta implementación requiere de experimentación y análisis del rendimiento general del sistema.

- **Mejorar colaborativamente**: Una vez que Kanban ya está implementado y el foco de atención recae en el flujo de trabajo, el carácter de mejora incremental del método vuelve a salir a la luz. Es a través de la discusión, análisis y puesta en marcha de nuevas ideas que el equipo puede encontrar bloqueos en el proceso y adaptarse a cambios rápidamente.

1.2.4. Herramientas de Kanban

Para poder hacer uso de los principios y aplicar correctamente las prácticas del método, existen una serie de herramientas y artefactos comunes dentro de la metodología Kanban.

INSERTAR UNA IMAGEN DE UN TABLÓN KANBAN COMPLETO

Como se puede observar en la figura (), muchos de los elementos visibles ya habían sido referenciados anteriormente. Sin embargo, se procederá a dar una explicación detallada de lo que representan y qué rol cumplen:

PONER LA REFERENCIA A LA IMAGEN DEL TABLÓN KANBAN

- **Tablón**: Como núcleo de la visualización del proceso está el tablón kanban (nótese la diferencia en la capitalización del nombre). Dichos tablones contienen las listas de tareas a realizar y permiten observar el flujo de trabajo del equipo a través de columnas y cartas. Cada grupo deberá especificar su propio tablón kanban en función de su proceso de desarrollo y es a través de éste que se podrán calcular las métricas, encontrar los ciclos de retroalimentación y determinar qué se debe mejorar para maximizar la eficiencia.
- **Listas o Columnas**: Para poder visualizar correctamente el flujo de trabajo dentro del tablón kanban se hacen uso de listas de tareas o columnas. Dichos elementos representarán los pasos que debe seguir una tarea para pasar por todo el flujo desde que se plantea como labor hasta que se finaliza. En el método se dice que las tareas son *tiradas* a la siguiente lista una vez que se cumplen las condiciones de finalización de la columna en la que se encuentra. Entre los tipos de columnas que se pueden encontrar están:
  - **Backlog**: Contiene la lista de tareas que se plantean realizar, pero aún no se han empezado o que pueden terminar siendo descartados. Éstas son las ideas de lo que se quiere desarrollar, pero que aún no se han materializado en labor. Dichos elementos deberán estar enfocados a contribuir con los objetivos del producto a implementar.
  - **Terminado**: Contiene la lista de tareas que se han dado por finalizadas en su totalidad. Esta columna es la última del tablón para el equipo y aquí es donde

estarán todas las tareas completadas una vez que han pasado por todo el flujo de desarrollo.

- **Listas intermedias**: El tablón kanban, al depender mucho proceso del equipo y no poder ser estandarizado, hará uso también de distintas columnas intermedias entre el *backlog* y la columna de tareas finalizadas. Dichas listas serán representativas del flujo de trabajo que sigue el grupo como tal para desarrollar el proyecto. Tipicamente se pueden encontrar columnas como: *a diseñar*, *a empezar*, *en progreso*, *en verificación*, entre otras.

- **WiP**: Con el fin de aplicar las prácticas de limitación y manejo del flujo de trabajo, a cada columna se le asigna un límite de tareas que puede contener en un determinado momento. Si dicha lista llega a su límite, no se podrán traer tareas a ella hasta que se liberé de los labores que contiene. Normalmente dicho límite se escribe al lado del nombre de su respectiva columna.
- **Cartas**: Así como el tablón kanban contiene listas de trabajo, dichas columnas también contienen unos elementos conocidos como *cartas*. Éstas representan las actividades y trabajos a realizar con el fin de avanzar con proyecto, siempre teniendo en cuenta que dichos elementos deben aumentar el valor del producto y tienen que ser significados para su finalización.

1.2.5. Definiendo los criterios de finalización

1.2.6. Métricas de flujo

TRADUCIR Y FINALIZAR

- **Trabajo en progreso**: Es el número de elementos o cartas de trabajo iniciados pero no finalizados.
- **Rendimiento**: Es el número de elementos de trabajo finalizados por unidad de tiempo.
- **Edad del elemento de trabajo**: Es el tiempo transcurrido desde que se inició el trabajo hasta el momento actual.
- **Tiempo de ciclo**: Es la cantidad de tiempo transcurrido desde que se inicia un trabajo y se termina.

1.2.7. Justificación de la metodología

Por último, la gestión del proyecto se realizará siguiendo la metodología de desarrollo ágil Kanban para permitir construir de manera iterativa las distintas funcionalidades del proyecto. Hacer uso del método Kanban para el desarrollo y gestión del proyecto.

No se puede definir una estimación de tiempo, por lo que se deberá usar una metodología ágil

1.2.8. Trello como herramienta de gestión

Once the starting format of the Kanban board is agreed, the first and almost pivotal decision to be made is whether to go for a physical board or an electronic one. Both have their pros and cons and there may be working practices that guide the final decision. A virtual board

can't be beaten for accessibility and ease of sharing, as you're never more than a smart phone or iPad away. But in our opinion the most important thing is for the board to be highly visible, and nothing can beat a physical board for that. A high-profile, physical board has an almost magical quality, like a fireplace in a huge front room, and draws people in. To start with it's more about curiosity, yet after a short while it becomes a centrepiece and focus for team activities. Work is planned, prioritised and progressed around the board. A physical board is also guaranteed to generate huge interest in unlikely places. Senior management love the visibility of a board, so expect a visit from the CEO or Finance Director within a week. For once they'll see what's really going on in the organisation without quizzing middle management or ploughing through turgid weekly reports.

Despite our absolute preference for an old-fashioned physical board, there are times when an electronic board either makes more sense or is even the only viable option. When individuals are regularly on the move or if the team is split over different locations, there are insurmountable physical issues to deal with and a tech option become more attractive. But before giving up on having a physical board think carefully, especially when trialling agile for the first time. A tech alternative will work well enough from a functional perspective but is far less visible and engaging, so many soft benefits will be lost. Don't go down that route just because members of the team occasionally work from home. Don't throw the baby out with the bathwater. If an electronic board is the only workable solution, consider driving it from a physical source - start with a wall and duplicate. The overhead of keeping two boards in sync will be offset by the benefits of having a real board. But when all else fails there are plenty of electronic options with good coverage across the main devices. Some are completely free and all the rest offer a trial period, so try before you buy.

Uno de los artefactos es el tablón como tal. Aquí se define Trello como tablón online/virtual

Definición de Trello y un poco de transfondo

Se sacará la información de aquí: Brechner, 2015

1.2.8.1. Significado de las listas

Uno de los artefactos son las listas. Aquí se definen las escogidas

- **Backlog:** Tareas de la iteración en espera de ser empezadas
- **To Do:** Tareas a realizarse
- **Doing:** Tareas que se están realizando
- **Testing:** Tareas que se están evaluando antes de darse por completadas
- **Done:** Tareas de la iteración terminadas
- **Approved:** Tareas discutidas y aprobadas

1.2.8.2. Formato de las cartas

Aquí se definen las cartas que son otro artefacto de Kanban. Conviene mejor usar capturas

1.2.8.3. Significado de etiquetas

Un añadido que se puede utilizar son las etiquetas como artefacto. Aquí se definen

- **Bug:** Cuando se detecta un error y se debe arreglar
- **Bloqueado:** Cuando una tarea no se puede completar debido a otras circunstancias
- **Pendiente:** Abreviado de "Pendiente de Retroalimentación". Indica que esta tarea debe discutirse en una reunión
- **No aceptada:** Indica que la tarea no ha sido aceptada para continuar en la siguiente iteración y debe retrabajarse, cambiarse o descartarse.

1.2.8.4. Criterios de movimiento de listas

Se comentó que hay que definir el concepto de "finalizado" para cada lista. Aquí se hace eso

1.3. Estimación de dedicaciones y tareas

No se puede predecir porque usamos Kanban. Sólo conocemos las fechas finales de entrega.

Paquete	Descripción	Estimación (Horas)
Ge.P	Gestión - Planificación	23
Ge.S	Gestión - Seguimiento	20
FeI	Formación e Investigación	46
Pr.Me	Proyecto - Memoria	46
Pr.Fr.Le	Proyecto - Framework - Lenguaje	44
Pr.Fr.Co	Proyecto - Framework - Core	48
Pr.Fr.Cli	Proyecto - Framework - CLI	21
Pr.Ma	Proyecto - Manual	11
IyP	Instalaciones y Paquetes	11
Contención	Horas de contención	30
	Total	300

Tabla 1.1: Estimación de tiempos de dedicación por paquetes de trabajo

Explicar que contención no es un paquete de trabajo, pero que es una parte importante de la estimación

1.3.1. Descripción de tareas a realizar

Aquí irán todas las tareas realizadas al final del proyecto. Esto no es fijo, surgirán más tareas a lo largo del proyecto (lo cual es aceptable porque la metodología es ágil)

1.3.1.1. Gestión

Planificación

1. Definición de la metodología de gestión.
2. Generación de la planificación inicial.
3. Automatización de herramientas de gestión con Trello.
4. Cambios y actualizaciones de la planificación.

Seguimiento y Control

1. Recogida de información sobre el desarrollo del proyecto
2. Contraste de la información de seguimiento con los planes, identificación de desviaciones significativas y actuación ante riesgos emergentes.
3. Preparación de documentos de cara a la próxima reunión.
4. Reuniones de fin de iteración

1.3.1.2. Formación e Investigación

1. Investigación y formación sobre la metodología Kanban.
2. Profundización sobre conceptos de simulación de sistemas.
3. Profundización sobre conceptos de compilación.
4. Investigación y formación sobre herramientas de desarrollo de CLIs.
5. Investigación y formación sobre generación de paquetes instalables para Python.
6. Investigación y formación sobre funciones de alto nivel y paquetes relacionados de Python.
7. Investigación sobre lenguajes de modelado y simulación.
8. Investigación sobre antecedentes del proyecto.
9. Exploración de alternativas al módulo de lenguaje.

1.3.1.3. Producto

Memoria

1. Diseño y preparación de la estructura

2. Redacción del resumen
3. Redacción de la introducción
4. Redactar contexto
5. Redacción de la gestión del proyecto
6. Redacción del marco teórico
7. Redacción del desarrollo
8. Redacción de conclusiones
9. Preparación e inclusión de referencias bibliográficas
10. Redacción de anexos
11. Validación y corrección de los índices
12. Maquetación y diseño de la memoria

Framework

- **Lenguaje:**
  1. Diseño del léxico del lenguaje.
  2. Diseño de la sintaxis del lenguaje.
  3. Diseño de la semántica del lenguaje.
  4. Diseño de módulos de cara a la implementación del lenguaje.
  5. Implementación del analizador léxico.
  6. Implementación del analizador sintáctico.
  7. Implementación de la tabla de símbolos.
  8. Implementación del análisis semántico.
  9. Documentación de los ficheros del lenguaje.
  10. Pruebas y verificación del lenguaje.
- **Núcleo:**
  1. Diseño de los módulos y paquetes del núcleo.
  2. Implementación de la estructura de datos para almacenar eventos.
  3. Implementación del reloj y el proceso de temporización de sucesos.
  4. Implementaciones de los procesos de inicialización y finalización.
  5. Implementación del generador de informes final.
  6. Implementación de distintos generadores de datos aleatorios.

7. Implementación de la inicialización de las variables globales, variables de entrada, contadores estadísticos y medidas de rendimiento para cualquier modelo.
8. Implementación de la inclusión de nuevos tipos de eventos y sus funciones de ejecución correspondientes.
9. Implementación de distintas estrategias de generación y eliminación de eventos dentro de la lista de sucesos.
10. Implementación de la especificación de una función de parada definida por el usuario.
11. Implementación de la funcionalidad de configuración de la simulación.
12. Documentación de los ficheros del núcleo.
13. Pruebas y verificación del núcleo.

- **CLI:**
  1. Diseño de las funcionalidades de la CLI.
  2. Implementación del generador de proyectos.
  3. Implementación del gestor de configuraciones del proyecto.
  4. Implementación del lanzador de modelos de simulación.
  5. Documentación de los ficheros de la CLI.
  6. Pruebas y verificación de la CLI.

Manual

1. Diseño y preparación de la estructura
2. Redacción de la introducción al producto
3. Especificación de dependencias y requisitos
4. Redacción del proceso de instalación
5. Redacción de la explicación del módulo del lenguaje
6. Redacción de la explicación del núcleo del framework
7. Redacción de la explicación de la CLI
8. Generación de ejemplos de uso
9. Preparación e inclusión de referencias bibliográficas
10. Redacción de anexos
11. Validación y corrección de los índices
12. Maquetación y diseño del manual

1.3.1.4. Instalación y Paquetes

1. Preparación de paquetes para LaTeX.
2. Preparación del entorno de trabajo.
3. Instalación de paquetes para el desarrollo del lenguaje.
4. Instalación de paquetes para el desarrollo del núcleo.
5. Instalación de paquetes para el desarrollo de la CLI.

1.4. Análisis de riesgos y viabilidad

1.4.1. Análisis de riesgos

1.4.1.1. Métricas

Probabilidad	Valor	Impacto	Valor
Nada probable	0.10	Muy bajo	0.05
Poco probable	0.30	Bajo	0.10
Probable	0.50	Medio	0.20
Muy probable	0.70	Alto	0.40
Casi probable	0.90	Muy alto	0.80

Tabla 1.2: Métricas para el análisis de riesgos

1.4.1.2. Tabla de riesgos

Nº	Descripción	Probabilidad	Impacto	Respuesta
1	Mala gestión del tiempo	Probable	Alto	Replanificar + hacer uso de la agilidad de la metodología
2	No entender la metodología de gestión y desarrollo (Kanban)	Probable	Alto	Profundizar en la formación de la metodología y generar cambios en el seguimiento y control

Continua en la siguiente página

Continuación de la página anterior

Nº	Descripción	Probabilidad	Impacto	Respuesta
3	Pérdida de información (se solventa con gestión de versiones + github)	Nada probable	Muy alto	Hacer uso de control de versiones para guardar todos los datos. En este caso GitHub.
4	Problemas de comunicación (por problemas de conexión, enfermedad, etc.)	Poco probable	Bajo	Replanificar reuniones?
5	Problemas con la integración de herramientas de desarrollo	Muy probable	Muy alto	Buscar alternativas en caso de no poder integrar las herramientas o profundizar en la formación de esta integración para evitar que ocurra
6	Análisis incorrecto de los usuarios finales del producto (Pensar en el lenguaje, por ejemplo).	Probable	Alto	Reanalizar los perfiles de usuarios objetivos y replantear los productos
7	Encontrado un proyecto idéntico (riesgo positivo (o se aceptan o se explotan))	Poco probable	Medio	Buscar una forma de hacer uso de éste, ya sea para integrarlo en el proyecto o para usarlo como formación.

Tabla 1.3: Riesgos del proyecto

Tenemos que diferenciar entre riesgos externos e internos

Hay riesgos negativos (que afectan malamente al proyecto) y riesgos positivos (que afectan positivamente al proyecto)

1.4.2. Análisis de viabilidad

Conocimientos suficientes de compilación, por tanto viable

Conocimientos suficientes en simulación de sistemas, por tanto viable

Viendo los antecedentes, considero que el proyecto es viable

Matriculado en 4 asignaturas, por lo que la calidad del proyecto podría verser afectada, por tanto no tan viable

Periodo vacacional de Semana Santa se puede usar para trabajar, por tanto viable

Debido a programa de intercambio, termino las clases en junio, por lo que podría no ser viable en cuanto a tiempo

Los riesgos más peligrosos son los relacionados a la falta de tiempo, por tanto podría no ser viable

1.5. Sistema de información y comunicaciones

1.5.1. Sistema de Información

1.5.1.1. Estructura

Aquí debemos explicar la estructura

TFG: Directorio principal del proyecto. Se divide en:

- **TFG.Proyecto:** Aquí residen los entregables del proyecto.
  - **Memoria:** Los ficheros relacionados a la memoria
  - **Framework:** Los ficheros relacionados al framework
    - **Lenguaje:** Los ficheros relacionados al lenguaje del framework
    - **Core:** Los ficheros relacionados al núcleo del framework
    - **CLI:** Los ficheros relacionados al CLI del framework
  - **Manual:** Los ficheros relacionados al manual
- **TFG.Gestión:** Aquí residen los ficheros relacionados a la gestión del proyecto
  - **Planificación:** Los ficheros relacionados a la planificación
  - **Seguimiento:** Los ficheros relacionados al seguimiento y control
  - **Actas:** Las actas de cada reunión realizada

EXISTE LA NECESIDAD DE CAMBIAR DE UN CLI A UNA GUI

1.5.1.2. Copias de seguridad

Usamos GitHub para control de versiones y copias de seguridad

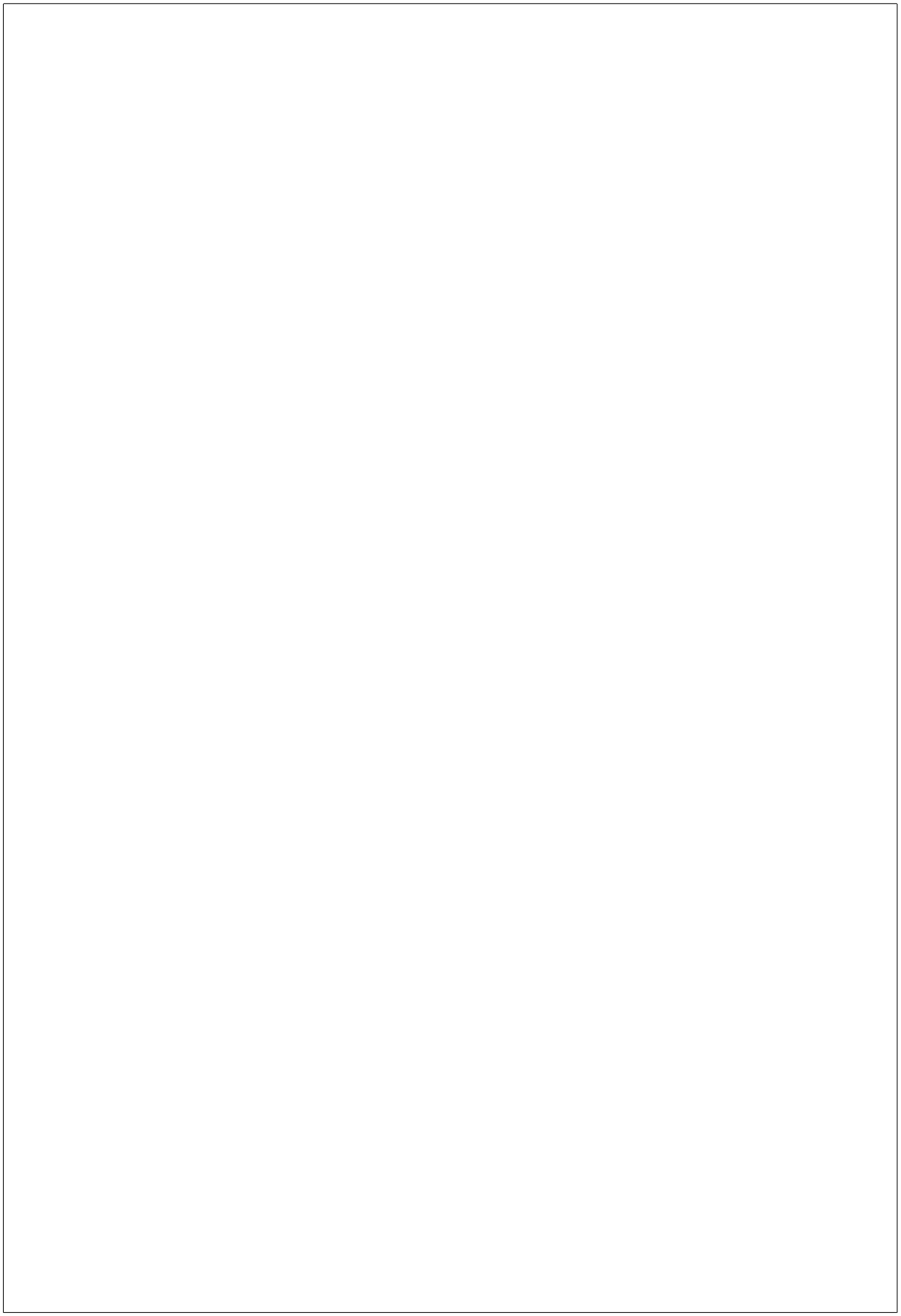
Todo está dividido en varios módulos y submódulos.

Los enlaces a todo están en: https://github.com/brodriguez059/TFG

Usamos Drive para generar las actas de fin de iteración

1.5.2. Comunicaciones

- (Reuniones a través de Webex)
- (Comunicaciones para responder dudas a través de email)



Bibliografía

Buss, A. (1996). Modeling with event graphs. *Winter Simulation Conference*, 153-160. https://doi.org/10.1109/WSC.1996.873273

Maria, A. (1997). Introduction to modeling and simulation. *Proceedings of the 29th conference on Winter simulation - WSC '97*. https://doi.org/10.1145/268437.268440

Perros, H. G. (2009). *Computer simulation techniques: the definitive introduction!*

Anderson, D. J. (2010). *Kanban*. Blue Hole Press.

Banks, J. (2010). *Discrete-event system simulation*. Prentice Hall.

Wainer, G. (2011). *Discrete-event modeling and simulation: theory and applications*. CRC Press.

Stellman, A. (2014). *Learning agile*. O'Reilly Media.

Brechner, E. (2015). *Agile Project Management with Kanban*. Microsoft Press.

Cole, R., & Scotcher, E. (2015). *Brilliant Agile project management : a practical to using Agile, Scrum and Kanban*. Pearson.

MATLAB. (2017a). Understanding Discrete Event Simulation, Part 1: What Is Discrete Event Simulation.

MATLAB. (2017b). Understanding Discrete Event Simulation, Part 2: Why Use Discrete Event Simulation.

MATLAB. (2017c). Understanding Discrete Event Simulation, Part 3: Leveraging Stochastic Processes.

MATLAB. (2017d). Understanding Discrete Event Simulation, Part 4: Operations Research.

MATLAB. (2017e). Understanding Discrete Event Simulation, Part 5: Communication Modeling.

Project Management Institute. (2017). *A guide to the project mangement body of knowledge (pmbok(r) guide)-sixth edition*.

Vacanti, D. S. (2020). The Kanban Guide. https://kanbanguides.org/wp-content/uploads/2020/07/Kanban-Guide-2020-07.pdf