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

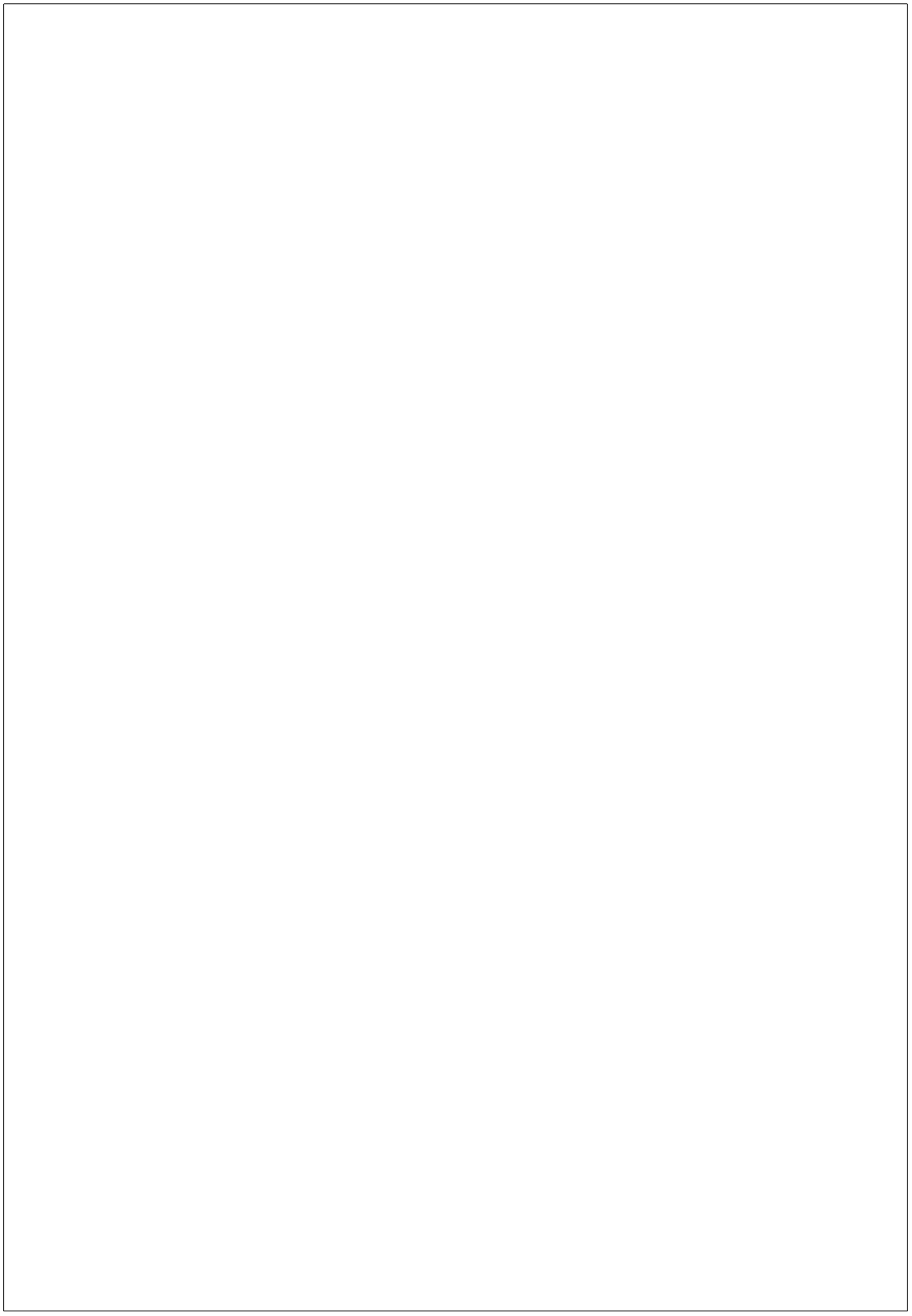
# Transpilador de un lenguaje de modelado personalizado de sistemas de simulación dinámicos discretos a Python

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

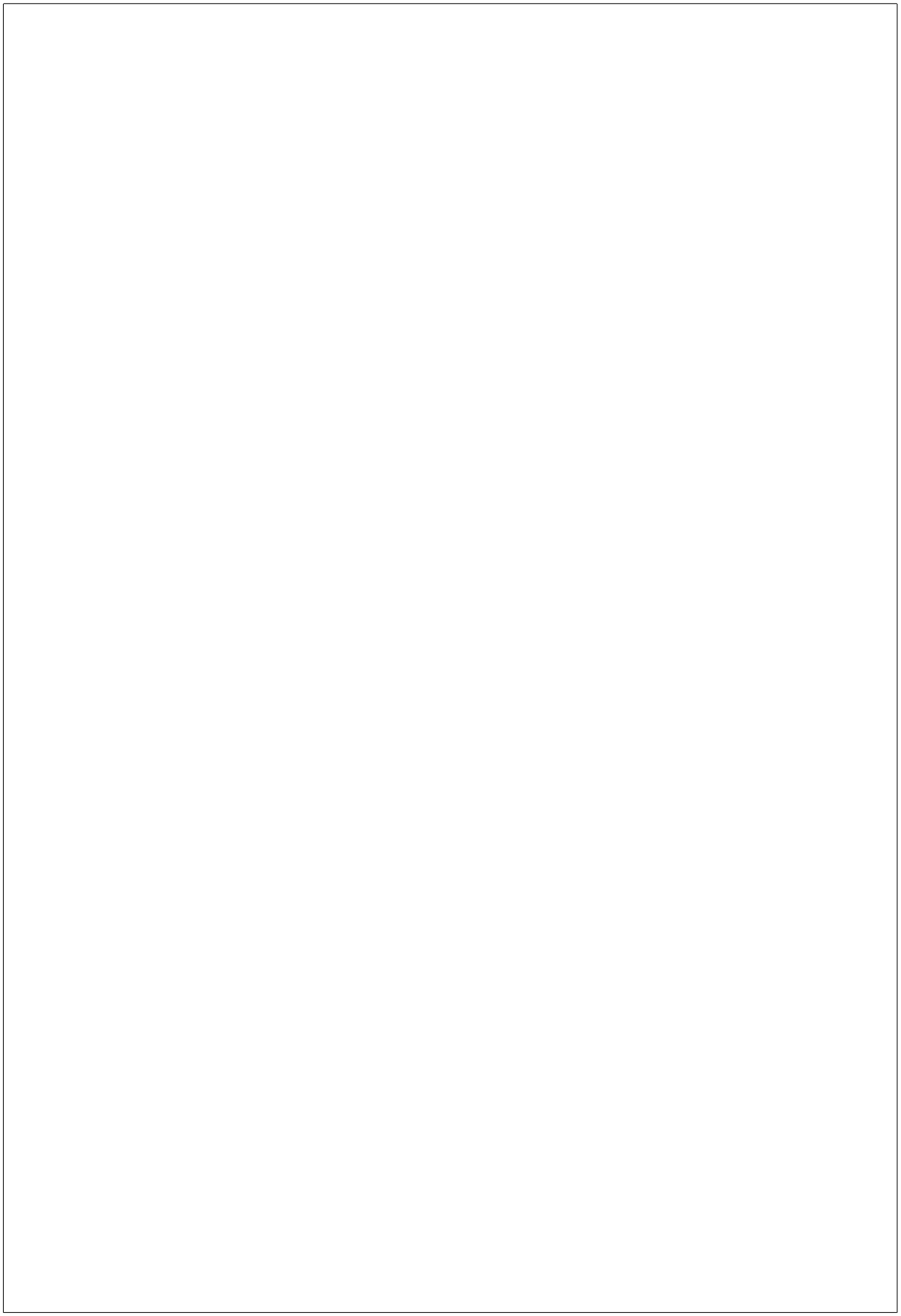
Autor

*Baldwin David Rodríguez Ponce*

Directora

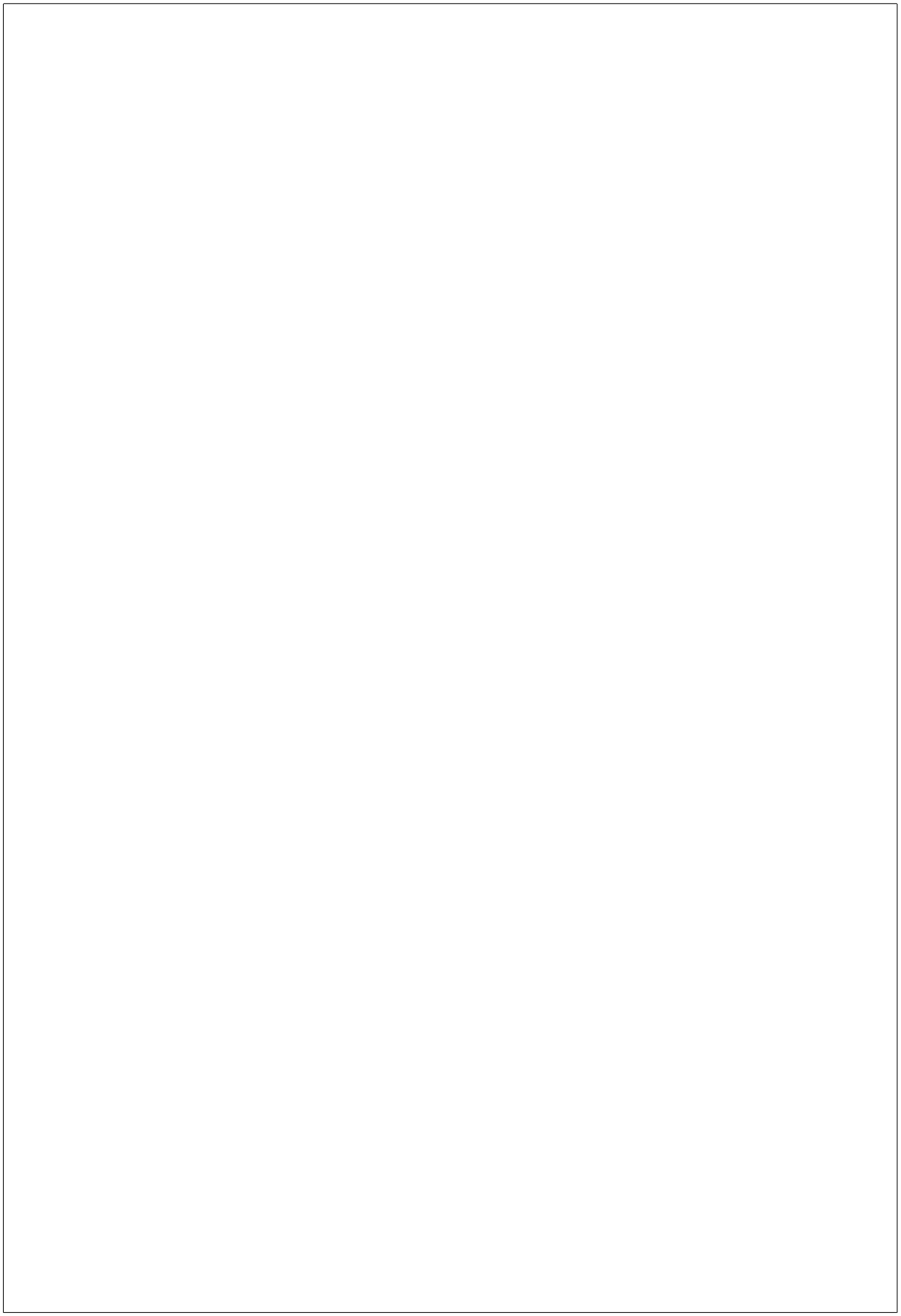
María Begoña Losada Pereda

BORRADOR. Nota: Márgenes del documento dibujados explícitamente.



# Tabla de contenidos

# Índice de figuras



# CAPÍTULO 1

# Contexto

## 1.1. Motivación

En el área de simulación de sistemas, una familia de modelos son los “modelos dinámicos discretos”, siendo estos “dinámicos” porque su comportamiento cambia en el tiempo y “discretos” porque estos cambios ocurren en instantes determinados en vez de ocurrir continuamente. Dichos sistemas se suelen modelar usando un diseño “basado en eventos”, el cual nos indica que los cambios del modelo se deben considerar como “eventos” que pueden ocurrir y dar lugar a otros eventos. Para ello, es común generar un “grafo de sucesos” que represente todos los eventos posibles que pueden ocurrir en el sistema y las relaciones que tienen entre estos.

Siento que estás repitiéndote mucho, tienes que cambiar la calidad de este discurso

Podríamos decir que cada modelo tendrá sus peculiaridades y diferencias específicas a la hora de generar su implementación. Sin embargo, se da el hecho de que todos los modelos de esta familia comparten elementos en común independientemente del sistema a simular: el reloj de la simulación, el temporizador de eventos, la lista de sucesos, entre otros. Por tanto, podemos abstraer el desarrollo de estos programas de tal forma que el desarrollador sólo deba encargarse de implementar todo aquello que sea único del modelo, quitándole así la responsabilidad de generar los elementos en común y agilizando el desarrollo en el proceso.

Tomando en cuenta que la principal herramienta de diseño de estos modelos serán los grafos de sucesos, nos encontraremos con el hecho de que éstos siguen una estructura representable a través de una gramática independiente de contexto. Por tanto, es posible generar una serie de analizadores léxico, sintáctico y semánticos que nos permitan procesar un lenguaje formal a otro código fuente. Por esta razón hemos considerado una buena opción el uso de herramientas de generación de compiladores como lo son Flex y Bison. Vemos que es posible crear estos analizadores de forma que este hipotético nuevo lenguaje sea traducido a código Python.

## 1.2. Antecedentes

### 1.2.1. Software para la construcción automática de programas de Simulación Continua

### 1.2.2. Lenguajes de simulación

Mencionar que hay dos categorías de lenguajes de programación para desarrollar los simuladores: específicos y generales. El nuestro será específico.

Aquí mencionaré algunos lenguajes de programación creados con el fin de desarrollar simuladores

### 1.2.3. Software de simulación

Aquí mencionaré algunas aplicaciones creadas para desarrollar simulaciones (como Arena)

# CAPÍTULO 2

# Gestión del proyecto

## 2.1. 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.

### 2.1.1. Objetivos

#### 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.

#### 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.

### 2.1.2. Requisitos

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

### 2.1.3. EDT

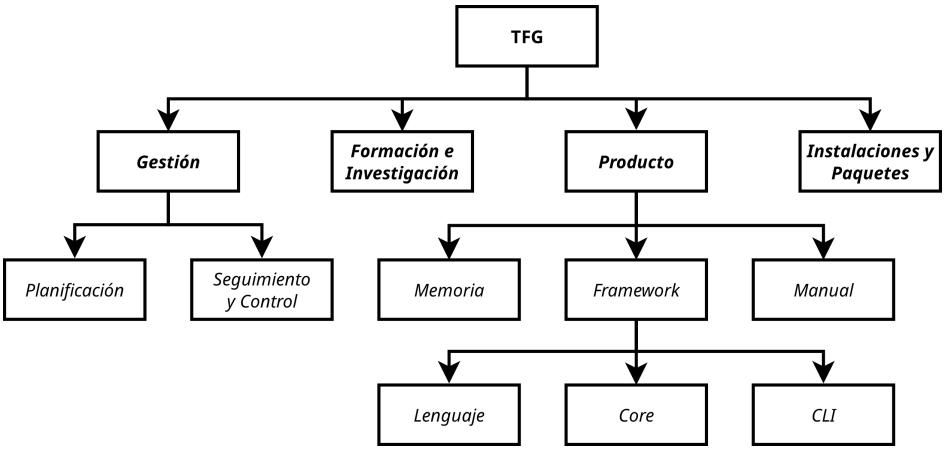



**Figura 2.1:** Esquema de Descomposición de Trabajo del Proyecto

#### 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 trabaj 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.

#### 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.

**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.

**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.

## 2.2. Metodología

Explicar Kanban

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.

Desglosar más lo que es Kanban

### 2.2.1. Trello como herramienta de gestión

**Significado de las listas**

- **Backlog:**
- **To Do:**
- **Doing:**
- **Testing:**
- **Done:**
- **Approved:**

**Significado de etiquetas**

- **Bug:**
- **Bloqueado:**
- **Pendiente:**
- **No aceptada:**

**Formato de las cartas**

## 2.3. Estimación de dedicaciones y tareas

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

### 2.3.1. Estimación de dedicación

| Paquete | Nombre | 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 | Total | 300 |

**Tabla 2.1:** 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

### 2.3.2. 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)

**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

**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.

**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

**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.

## 2.4. Análisis de riesgos y viabilidad

### 2.4.1. Análisis de riesgos

#### Métricas

| Probabilidad | Valor | Impacto | Valor | Nivel de Riesgo | Probabilidad x Impacto |
|---|---|---|---|---|---|
| Nada probable | 0.10 | Muy bajo | 0.05 | Muy bajo | De 0 a 0.04 |
| Poco probable | 0.30 | Bajo | 0.10 | Bajo | De 0.05 a 0.09 |
| Probable | 0.50 | Medio | 0.20 | Medio | De 0.10 a 0.29 |
| Muy probable | 0.70 | Alto | 0.40 | Alto | De 0.30 a 0.49 |
| Casi probable | 0.90 | Muy alto | 0.80 | Muy alto | Mayor a 0.50 |

**Tabla 2.2:** Métricas para el análisis de riesgos

#### Tabla de riesgos

| Nº | Descripción | Probabilidad | Impacto | Respuesta |
|---|---|---|---|---|
| 1 | Mala gestión del tiempo (falta de tiempo + pérdida de 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 |
| 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. |

Continua en la siguiente página

Continuación de la página anterior

| Nº | Descripción | Probabilidad | Impacto | Respuesta |
|---|---|---|---|---|
| 4 | Problemas de contacto con [directora] (por problemas de conexión, enfermedad, etc. Tanto en el caso del autor como en el caso de la directora) | 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 (Piensa 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 2.3:** 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)

### 2.4.2. Análisis de viabilidad

Viendo los antecedentes, considero que el proyecto es viable

## 2.5. Sistema de información y comunicaciones

### 2.5.1. Sistema de Información

#### 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

### 2.5.2. Comunicaciones

- [Reuniones a través de Webex]
- [Comunicaciones para responder dudas a través de email]

# CAPÍTULO 3

# Marco Teórico

## 3.1. Simulación

A lo largo del documento hemos estado mencionando la simulación de sistemas, pero no hemos definido exactamente qué es.

La simulación es una potente técnica de resolución de problemas

Orígenes: La teoría de muestreo estadístico y el análisis probabilístico de complejos sistemas físicos

Punto en común: uso de número aleatorios y muestreo aleatorio para aproximar un resultado de una solución

Primera aplicación: Por Von Neuman y Ulam, durante la segunda guerra mundial, estudio de problemas de difusión aleatoria de neutrons, en conexión con el desarrollo de la bomba atómica

El proyecto era alto secreto, se le dio un nombre clave: Monte Carlo, en referencia al famoso casino de juego

El nombre persistió durante un tiempo como sinónimo de cualquier simulación pero hoy en día está restringido a una rama de la matemática experimental que trata con números aleatorios (y que puede relacionarse con lo que llamaremos modelos de simulación estáticos)

Mientras que el término simulación, o simulación de sistemas, se refiere a una técnica de análisis más extensa, aunque muy a menudo utiliza números aleatorios

La contribución individual más importante es la potencia de cálculo y velocidad de procesamiento de los ordenadores

Definición informal:

> Una simulación es la imitación del modo de funcionamiento u operación de un proceso o sistema del mundo real. La simulación involucra la generación de una historia artifical de un sistema, y la observación de esa historia artificial para obtener inferencias relativas a las características de funcionamiento de dicho sistema.

Cambios en esta redacción

Definición formal:

Nosotros hablaremos de una definición más exacta de “Simulación de Sistemas” y “Modelado” más adelante, pero por ahora podemos citar la definición encontrada en

> El proceso de diseñar un modelo lógico o matemático de un sistema real y realizar experimentos basados en el ordenador con el modelo, al objeto de descrbir, explicar o predecir el comportamiento del sistema real.

Cambios en esta redacción

Actualmente hay muchos tipos distintos de sistemas de simulación, como por ejemplo los “Modelos de Montecarlo” y “Modelos en ecuaciones diferenciales”. No obstante, el principal objeto de estudio de este proyecto será una categoría muy importante a la que se conoce como “Modelos de simulación dinámicos discretos”, especificamente aquellos que se pueden modelar a través de un diseño orientado a eventos.

Se hace uso de otras técnicas:

La modelización permite obtener una representación abstracta del sistema real

Las técnicas de programación de ordenadores: El programa de ordenador traduce el modelo a una forma operativa

La teoría de la probabilidad define las variables aleatorias del modelo, y ayuda a construir las historias artificaciones (generadores de datos) necesarios

La estadística ayuda en el diseño y análisis de experimentos a realizar para obtener respuestas

Los métodos heurísticos se emplean para conseguir buenas soluciones, sino óptimas

El modelado matemático o modelado analítico se aprovecha de las características del problema cuya respuesta se desea obtener para llegar a la mejor conclusión. Sin embargo, muchas veces nos encontraremos con problemas que no se pueden modelar analíticamente debido a su complejidad en tiempo o espacio. En estas situaciones, debemos hacer uso de técnicas de aproximación de resultados para dar con soluciones que, aunque no se pueden garantizar óptimas, sí que se pueden considerar lo suficientemente buenas. Una de estas técnicas vendría a ser la simulación de sistemas, la cual a través de modelado simbólico/lógico intenta reproducir el comportamiento de un determinado sistema con el fin de analizar una serie de resultados a escoger.

La simulación tiene sentido en la medida en que un ordenador permite plantear, describir y condificar modelo de grandes y complejos sistemas, que no se pueden resolver por el cálculo matemático estadístico de forma útil

Se abre la posibilidad de, al igual que por ejemplo, los físicos y biólogos, realizar experimentos "de laboratorio.[en] áreas donde tradicionalmente esto no se podía hacer

Esto representa una gran ventaja en cuanto a las posibilidades de controlar las condiciones de tales experimentos y en consecuencia incrementar la información obtenida de los mismos

### 3.1.1. Definiciones

## Modelo

Un modelo es una representación o una abstracción de un sistema con el propósito de estudiar tal sistema, pero que contiene sólo lo esencial del sistema real

Aquellos aspectos del sistema que no contribuyen de forma significativa al comportamiento del sistema no están incluidos en el modelo. Por tanto un modelo es: un substituto del sistema real, una simplificación del mismo

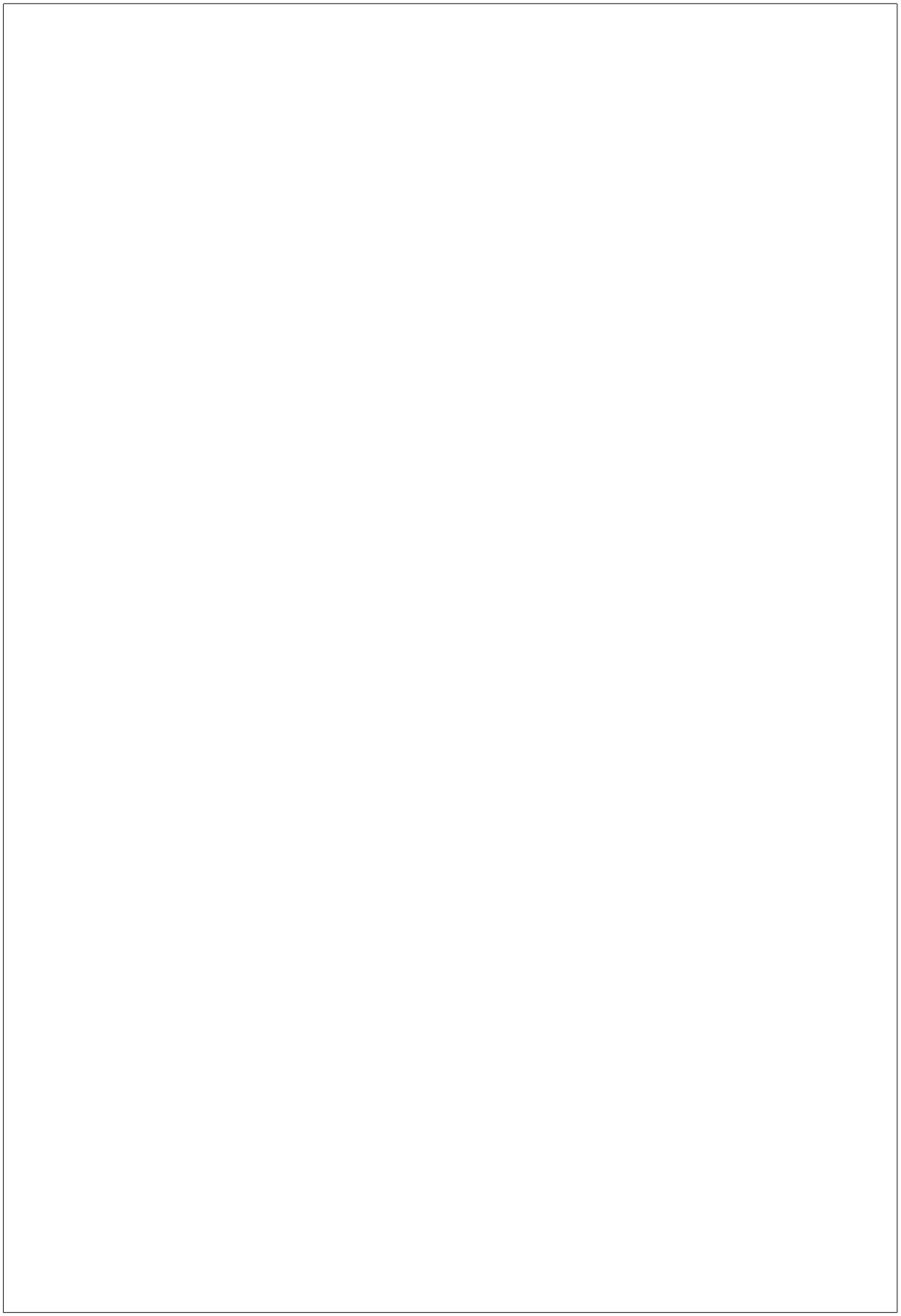
Un modelo probabilístico

## Sistema

### 3.1.2. Ventajas e inconvenientes

### 3.1.3. Diseño basado en eventos

### 3.1.4. Sistemas dinámicos discretos



# CAPÍTULO 4

# Desarrollo

## 4.1. Diseño

### 4.1.1. Lenguaje

#### Léxico

Tabla de tokens Autómata

#### Sintaxis + Semántica

Gramática Definición de atributos Descripción de abstracciones funcionales ETDS Tabla de símbolos

### 4.1.2. Núcleo

Diagrama de clases?

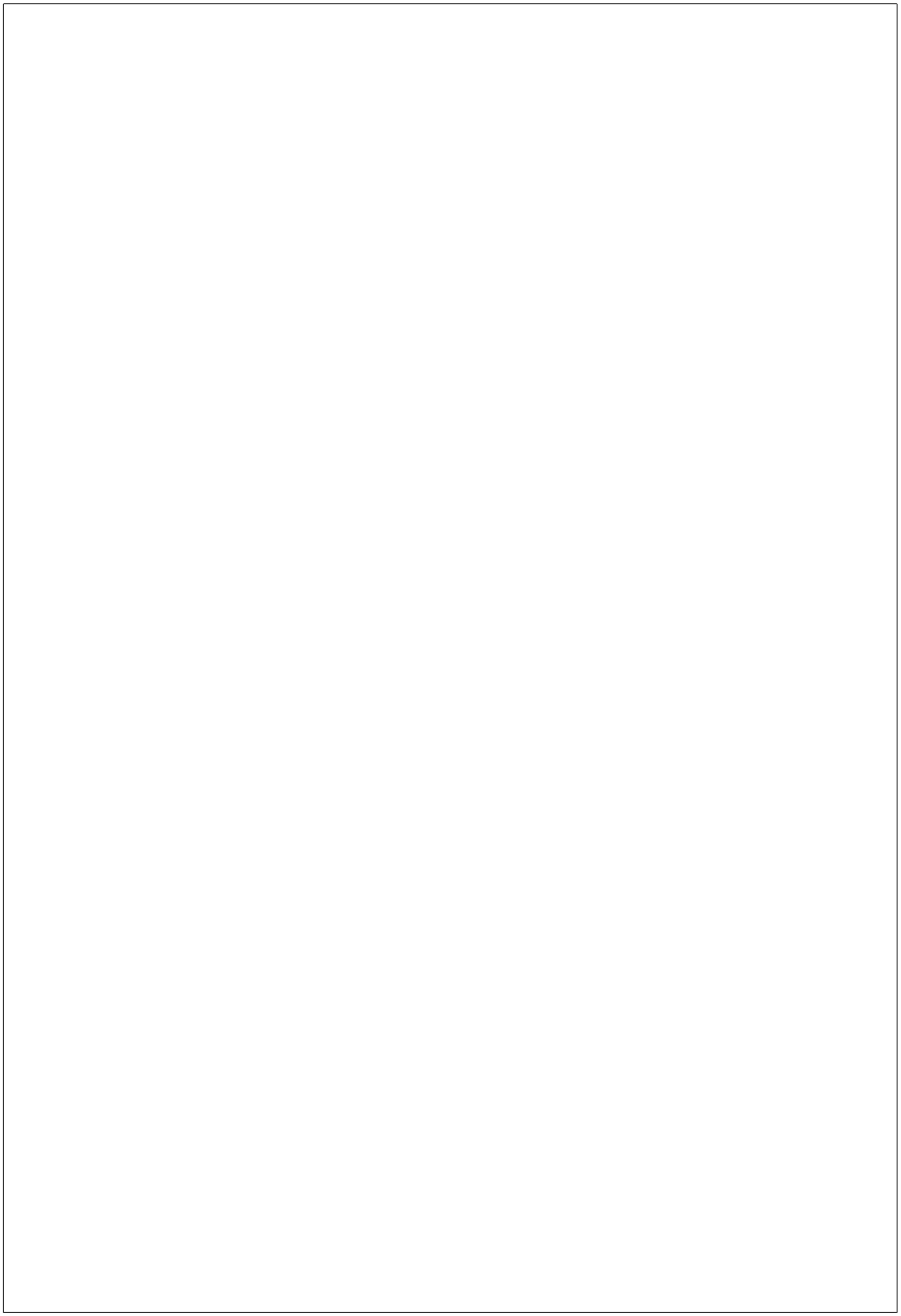
### 4.1.3. CLI

## 4.2. Implementación

### 4.2.1. Lenguaje

### 4.2.2. Núcleo

### 4.2.3. CLI



# 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!*

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

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

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.