



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.

--

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	v
1. Gestión del proyecto	1
1.1. Alcance	1
1.1.1. Objetivos	2
1.1.2. Requisitos	3
1.1.3. EDT	3
1.2. Metodología	4
1.2.1. Principios de Kanban	5
1.2.2. Prácticas de Kanban	7
1.2.3. Métricas de flujo	7
1.2.4. Herramientas de Kanban	8
1.2.5. Definiendo los criterios de finalización	8
1.2.6. Justificación de la metodología	8
1.2.7. Trello como herramienta de gestión	8
1.3. Estimación de dedicaciones y tareas	10
1.3.1. Descripción de tareas a realizar	11
1.4. Análisis de riesgos y viabilidad	14
1.4.1. Análisis de riesgos	14
1.4.2. Análisis de viabilidad	16
1.5. Sistema de información y comunicaciones	16
1.5.1. Sistema de Información	16
1.5.2. Comunicaciones	17
2. Marco Teórico	19
2.1. Simulación	19
2.1.1. Definiciones	20
2.1.2. Ventajas e inconvenientes	21
2.1.3. Diseño basado en eventos	21
2.1.4. Sistemas dinámicos discretos	21

Índice de figuras

1.1. EDT del Proyecto 3

--

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.

- **Núcleo:** Una serie de módulos que implementarán un microframework de simulación

Cambiar redacción

Aquí te falta la posibilidad de poner las cosas variables también

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:

- 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

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.

1.1.2. Requisitos

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

Requisitos funcionales

TERMINAR

Requisitos no-funcionales

TERMINAR

1.1.3. EDT

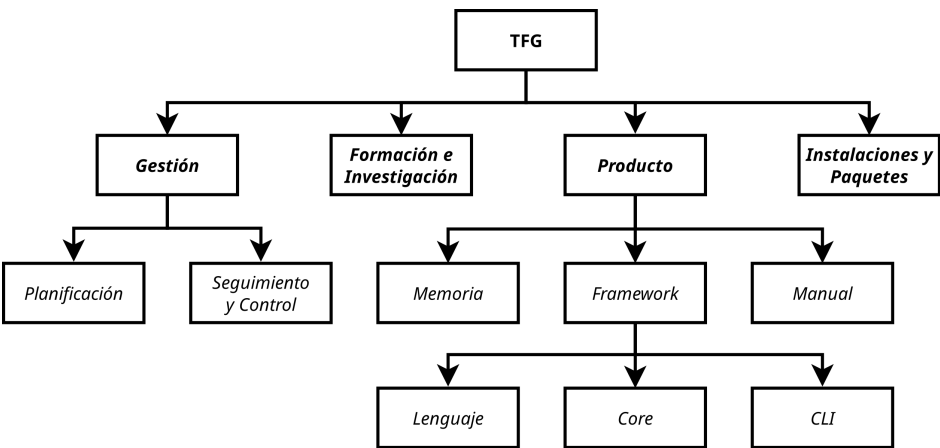


Figura 1.1: Esquema de Descomposición de Trabajo del Proyecto

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 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.

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.

1.2. Metodología

Definición de Kanban: “Kanban es un...”

Se sacará la información de aquí principalmente: Cole y Scotcher, [2015](#) Stellman, [2014](#)

Kanban's great for:

- Getting off to a low-risk, zero-cost, agile, fast start.
- Pinning down existing workflows and spotting glaring errors.
- Controlling multiple pieces of unconnected work.
- Keeping the numbers of jobs in play down to an acceptable level.
- Getting the team into an agile way of thinking.

However, there are subtle, yet important, differences with Kanban:

All work is of a similar size. It's better to have smaller stories of roughly equal size. Splitting large pieces of work down into smaller, similar sized packages has been proven to improve workflow and results in more predictive end-to-end cycle times. Comparing like-for-like can also aid the review process.

The backlog is refined more regularly. The Kanban backlog tends to be exceptionally dynamic especially in a support type environment. Backlogs in other agile environments are far from static but they're just a touch less zippy. It's not unusual to review a Kanban backlog on a daily basis.

Jobs are pulled not pushed. A Kanban team adopts a what's next policy rather than packaging up work into a connected delivery. The task assigned the highest priority is pulled into play when resource becomes available.

1.2.1. Principios de Kanban

- **Empieza con lo que tienes ya:** The thing about habitual problems is that they're habitual. When your team does something on a project that will eventually lead to bugs and missed deadlines, it doesn't feel like a mistake at the time. You can do all the root-cause analysis that you want after the fact; faced with the same choice, the team will probably still make the same decision. That's how people are. For example, let's say that a programming team always finds themselves delivering software to their users, only to repeatedly have awkward meetings where users can't find the features they think they were promised. Now, it's certainly possible that these developers are incredibly forgetful, and that they always forget one or two of the features that they discussed with the users. But it's more likely that they have a recurring problem with how they gather their requirements or communicate them to the users. The goal of process improvement is to find recurring problems, figure out what those problems have in common, and come up with tools to fix them. The key here is the second part of that sentence: figuring out what those problems have in common. If you just assume, for example, that a developer simply can't remember all of the things the users asked for, or that the users constantly change their minds, then you've effectively decided that the problems are unfixable. But if you assume that there's a real root cause

that's happening over and over again, then you stand a chance of finding and fixing it. That's where Kanban starts: taking a look at how you work today, and treating it as a set of changeable, repeatable steps. Kanban teams call steps or rules that they always follow policies. This essentially boils down to a team recognizing their habits, seeing what steps they take every time they build software, and writing all of those things down. Writing down the rules that a team follows can sometimes be tricky, because it's easy to fall into the trap of judging a team—or an individual team member—on the results of a project: if the project is successful, everyone on the team must be good at their jobs; if the project fails, they must be incompetent. This is unfair, because it assumes that everything in the project is within the control of the team. Lean thinking helps get past this by telling us to see the whole, which in this case means seeing that there's a bigger system in place. That's worth repeating: every team has a system for building software. This system may be chaotic. It may change frequently. It may exist mainly inside the heads of the team members, who never actually discussed a larger system that they follow. For teams that follow a methodology like Scrum, the system is codified and understood by everyone. But many teams follow a system that exists mainly as “tribal” knowledge: we always start a project by talking to these particular customer representatives, or building that sort of schedule, or creating story cards, or having programmers jump in and immediately start to code after a quick meeting with a manager.

- **Comprométete al cambio evolutivo e incremental:** This is the system that Kanban starts with. The team already has a way to run their project. Kanban just asks them to understand that system. That's what it means to start with what you do now. The goal of Kanban is to make small improvements to that system. That's what it means to pursue incremental, evolutionary change—and why Kanban has the practice improve collaboratively, evolve experimentally. In lean thinking, part of seeing the whole is taking measurements, and measurements are at the core of experimentation and the scientific method. A Kanban team will start with their system for building software, and take measurements to get an objective understanding of it. Then they'll make specific changes as a team—later in this chapter, you'll learn exactly how those changes work—and check their measurements to see if those changes have the desired effect.
- **Respetar el proceso, los roles, las responsabilidades y títulos actuales:** The Lean value of amplifying learning is also an important part of evolving the system that your team uses to build software. Throughout this book you've learned about feedback loops. When you collaborate to measure the system and evolve experimentally, those feedback loops become a very important tool for gathering information and feeding it back into the system; the Kanban practice of implementing feedback loops should make sense to you, and should help you to see how Kanban and Lean are closely linked. Amplifying learning also factors into the Kanban principle of initially respecting

current roles and responsibilities. For example, say that a team always starts each project with a meeting between a project manager, a business analyst, and a programmer. They may not have written down a rule for what goes on in that meeting, but you probably have a good idea of what goes on in it just from reading those job titles. That's one reason why Kanban respects current roles, responsibilities, and job titles—because they're an important part of the system. A common theme between all of these principles is that Kanban only works for a team when they take the time to understand their own system for building software. If there was one right way to build software, everyone would just use it. But we started this book by saying back in Chapter 2 that there is no silver bullet—there's no single set of “best” practices that will guarantee that a team builds software right every time. Even the same team, using the same practices, can have success with one project but fail miserably in the next one. This is why Kanban starts with understanding the current system for running the project: once you see the whole system, Kanban gives you practices to improve it.

1.2.2. Prácticas de Kanban

- **Definir y visualizar el flujo de trabajo:**
- **Limitar el trabajo en progreso:**
- **Manejar el flujo de trabajo:**
- **Hacer explícitas las políticas de proceso:**
- **Implementar ciclos de retroalimentación:**
- **Mejorar colaborativamente:**

1.2.3. Métricas de flujo

- **Trabajo en progreso:** The number of work items started but not finished (according to the DoW).
- **Rendimiento:** The number of work items finished per unit of time. Note the measurement of throughput is the exact count of work items.
- **Edad del elemento de trabajo:** The amount of elapsed time between when a work item started and the current time.
- **Tiempo de ciclo:** The amount of elapsed time between when a work item started and when a work item finished.

TRADUCIR
Y FINALI-
ZAR

1.2.4. Herramientas de Kanban

- **Tablón:** At the heart of the Kanban method is a deceptively clever tool: the Kanban board. Calling these boards a visual to-do list is an over-simplification but a decent starting point. The board is a graphic representation of the work to be done and the end-to-end flow from start to finish. The simplest and some argue the most pure Kanban board consists of just three columns: things to do, tasks in progress and finally work done. This simple format is universal and matches any project or corporate workflow.
- **Listas:** Keep it simple to start with and try out the four previously suggested stages: ideas, to do, doing and done. The demarcation between each status is clear and in turn generates the triggers for moving cards on:
 - **Backlog:** the maybe or maybe not stage when there's a question mark of any sort outstanding. At the very heart of the Kanban board are the to-do items also known as the backlog in various agile frameworks. These individual items are all delivery focused and must deliver business value directly or indirectly. For example, setting up a Kanban board is a legitimate item but a meeting to discuss the options is merely part of the main job. The tasks are business delivery focused and not centred on activities. If an item on the backlog does not contribute to business goals, it should be removed.
 - **Terminado:** totally complete with nothing more to do except reap the rewards or accept the gratitude.
 - **Listas intermedias:** To do, In progress, Testing...
- **Cartas:**

1.2.5. Definiendo los criterios de finalización

1.2.6. 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

1.2.7. 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 trans fondo

Se sacará la información de aquí: Brechner, [2015](#)

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

Formato de las cartas

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

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.

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)

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.

1.4. Análisis de riesgos y viabilidad

1.4.1. Análisis de riesgos

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

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
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
Continúa en la siguiente página				

Continuación de la página anterior

Nº	Descripción	Probabilidad	Impacto	Respuesta
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 verse 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

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

Aquí debemos explicar la estructura

- **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 CAM-
BIAR DE UN
CLI A UNA
GUI

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)

--

CAPÍTULO 2

Marco Teórico

2.1. 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 d

e 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 artificial 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

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 describir, explicar o predecir el comportamiento del sistema real.

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”, específicamente 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 artificiales (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 condicionar 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

2.1.1. 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

2.1.2. Ventajas e inconvenientes

2.1.3. Diseño basado en eventos

2.1.4. Sistemas dinámicos discretos

--

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.
- 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>

--
