UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

GLOBAL-MANAGER: Entrenando mediante un Juego Serio a los Jefes de Proyecto en los Desafíos del Desarrollo Global del Software

Rubén Márquez Villalta

julio, 2020

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de la Información

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

GLOBAL-MANAGER: Entrenando mediante un Juego Serio a los Jefes de Proyecto en los Desafíos del Desarrollo Global del Software

Autor: Rubén Márquez Villalta

Tutor: Francisco Pascual Romero Chicharro

Co-Tutor: Aurora Vizcaíno Barceló

julio, 2020

Tribunal:

Presidente: _____

Vocal: _____

Secretario: _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario
Fdo.:	Fdo.:	Fdo.:

Dedicatoria...

Resumen

En la actualidad, cada vez más empresas están introduciendo un nuevo modelo de desarrollo software, el cual resulta ser más des-localizado que el modelo convencional, donde los miembros del proyecto pueden estar en distintos países. Esta tendencia, llamada Desarrollo Global de Software (DGS), está creciendo rápidamente debido a la globalización, sin embargo, conlleva que aparezcan nuevos riesgos y desafíos en la gestión, los cuales pueden ser agrupados en tres bloques: Comunicación, Coordinación y Control. Es por esto, que es necesaria la existencia de Jefes de Proyectos preparados para afrontar y solventar los problemas que puedan ocurrir en este tipo de proyectos, por lo que se requiere contar con ciertas habilidades técnicas y no técnicas para llevar a cabo una correcta gestión del proyecto.

Por otro lado, últimamente, ha cobrado una gran importancia el uso de técnicas de Gamificación en el mundo de la enseñanza, en especial en el desarrollo de Juegos Serios para la formación y el entrenamiento de un conjunto de conocimientos y habilidades especificas. Por eso, en este proyecto se pretenderá llevar a cabo el desarrollo de un Juego Serio, titulado GLOBAL-MANAGER, para el entrenamiento de Jefes de Proyecto en habilidades necesarias para afrontar con éxito la gestión de un proyecto de software global. GLOBAL-MANAGER presentará a sus jugadores situaciones con desafíos que pueden tener lugar en este tipo de proyectos, y el jugador con el rol de Jefe de Proyecto tendrá que solventar dichas dificultades con el objetivo de finalizar correctamente la gestión de un proyecto global. Este juego contará con un módulo de Inteligencia Artificial con el fin de monitorizar las acciones del jugador y ajustar dinámicamente el desarrollo del mismo a los conocimientos aprendidos por el jugador.

Abstract

Nowadays, more and more companies are introducing a new model of software development, which turns out to be more de-localized than the conventional model, where the members of the project can be in different countries. This trend, called Global Software Development (GSD), is growing rapidly due to globalization, however, it brings new risks and challenges in management, which can be grouped into three blocks: Communication, Coordination and Control. For this reason, it is necessary to have Project Managers prepared to face and solve the problems that may occur in this type of projects, so it is required to have certain technical and non-technical skills to carry out a correct management of the project.

On the other hand, lately, the use of Gamification techniques has become very important in the world of education, especially in the development of Serious Games for the formation and training of a specific set of knowledge and skills. Therefore, in this project we intend to carry out the development of a Serious Game, entitled GLOBAL-MANAGER, for the training of Project Managers in skills needed to successfully manage a GSD project. GLOBAL-MANAGER will present to its players situations with challenges that can take place in this type of projects, and the player with the role of Project Manager will have to solve these difficulties in order to correctly finish the management of a global project. This game will have an Artificial Intelligence module in order to monitor the actions of the player and dynamically adjust the development of the game to the knowledge learned by the player.

AGRADECIMIENTOS

Rubén Márquez Villalta
Ciudad Real, 2020

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

ÍNDICE DE ALGORITMOS

CAPÍTULO 1

INTRODUCCIÓN

1.1. MOTIVACIÓN

En los últimos años, la Ingeniería del Software ha observado notables cambios a la hora de desarrollar proyectos software. Tradicionalmente, el modelo de desarrollo de software utilizado consistía en la coordinación de diferentes equipos de trabajo en un mismo edificio (*Desarrollo Colocalizado*), posteriormente, estos equipos de trabajo pasaron a organizarse entre diferentes edificios de una o varias ciudades, pero siempre centralizados en un mismo país (*Desarrollo Distribuido*). Sin embargo, en la actualidad y debido a la globalización, cada vez más compañías separadas geográficamente colaboran para desarrollar software hasta traspasar fronteras llegando a un nivel mundial, por lo que se ha evolucionado hacia un modelo de desarrollo mucho más globalizado y deslocalizado, conocido como *Desarrollo Global de Software* (DGS), o en inglés *Global Software Development* (GSD) [38].

El DGS está teniendo cada vez más aceptación entre los profesionales. Esta tendencia consiste en la colaboración entre diferentes equipos de desarrollo (fig. 1.1), los cuales se encuentran ubicados alrededor del mundo en diferentes ciudades, países y continentes. Estos grupos de trabajo pueden pertenecer a distintas organizaciones, pero trabajan conjuntamente en un mismo proyecto software. En el proyecto podrá existir tanto una comunicación *asíncrona* como *síncrona* entre los equipos de trabajo, lo cual dependerá de una serie de características del proyecto [39].

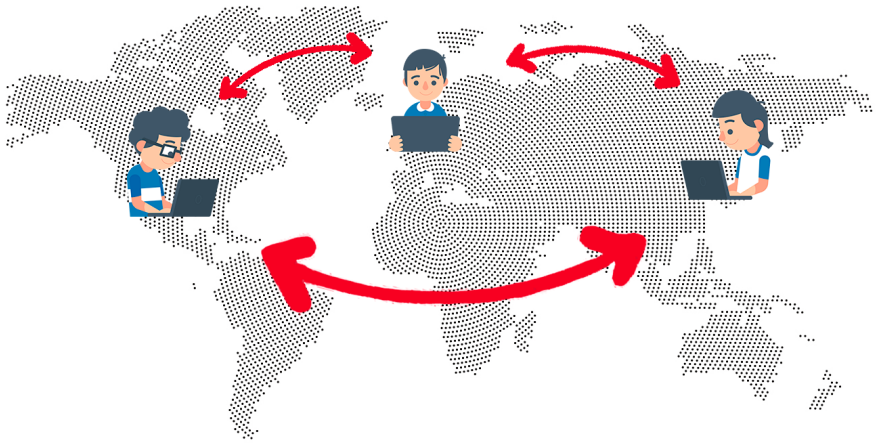

Figura 1.1: En el DGS diferentes equipos de desarrollo colaboran a nivel mundial en un mismo proyecto

Gradualmente, esta tendencia esta cogiendo cada vez más fuerza en el campo de la ingeniería del software, considerándose una norma en el desarrollo de software [8]. Esto es debido a que las

organizaciones pueden conseguir grandes beneficios utilizando este nuevo modelo de desarrollo, ya que la principal ganancia que se consigue con su uso es la reducción en el coste económico de los proyectos, debido a que se suelen buscar territorios donde la mano de obra cualificada es barata y fácilmente disponible [31]. Además, se pueden encontrar otros beneficios notables como pueden ser el acercamiento del desarrollo del software al cliente y al mercado local, la reducción del período necesario para el desarrollo del software al maximizar la productividad y la expansión hacia la inclusión de trabajadores mayormente cualificados en sus actividades de desarrollo [2].

Sin embargo, acompañando a las anteriores ventajas que se pueden conseguir con los proyectos DGS, existen una serie de inconvenientes, los cuales son causados, principalmente, a las diferencias existentes en este tipo de proyectos las cuales podemos dividir en cuatro clases: las *diferencias lingüísticas*, la *distancia geográfica*, la *diferencia cultural* y la coexistencia de diferentes *zonas horarias*; haciendo mucho más difícil el consenso y entendimiento común [31]. Estas diferencias acentúan la problemática de administrar y gestionar un proyecto software, apareciendo así los tres principales desafíos en la gestión de proyectos DGS (fig. 1.2), también llamado en [38] como las tres ces:

- Desafíos en la comunicación. Los equipos de trabajo deben mantener una comunicación adecuada y activa, con el fin de llevar a cabo un intercambio constante de información y conocimientos.
- Desafíos en la coordinación. Las tareas deben estar sincronizadas, para no sufrir retrasos y alcanzar objetivos e intereses comunes.
- Desafíos en el control. El proyecto debe ser gestionado constantemente y confirmar que se cumplen fechas de entrega, estándares, presupuestos, etc.; además de solventar posibles contratiempos que puedan ocurrir durante el ciclo de vida del proyecto.

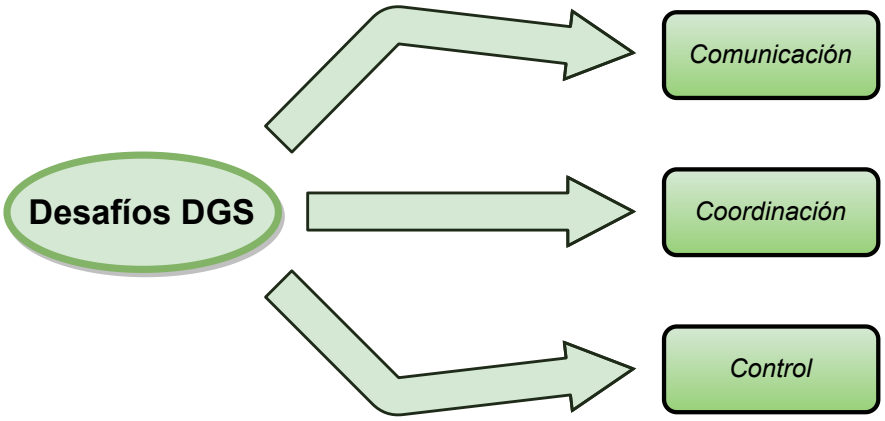

Figura 1.2: Los tres principales desafíos en la gestión de proyectos DGS

Estos inconvenientes y desafíos complican la gestión de este tipo de proyectos, lo que puede implicar en el retraso de tareas o incluso en la cancelación del mismo, ya que según la literatura, muchos proyectos DGS terminan fracasando. Según [27], la principal causa del fracaso de estos proyectos es debido a la imperfecta y dificultosa gestión de los mismos. Es por esto, que para conseguir los beneficios que nos ofrece el DGS es necesario que los jefes de proyecto posean grandes conocimientos y experiencia en la gestión de estos proyectos, además de contar con una serie de habilidades (no solo técnicas, sino también no técnicas), para hacer frente a los posibles contratiempos que puedan ocurrir en el ciclo de vida del proyecto y conseguir la finalización exitosa del mismo.

1.2. PROPUESTA

Existe una gran problemática con esta nueva tendencia de desarrollar software y un elevado número de proyectos terminan fracasando, y son escasos aquellos que consiguen finalizar exitosamente, obteniéndose así los beneficios que se consiguen frente al modelo de desarrollo tradicional. Esta situación se debe a que la educación en ingeniería del software no suele tener en cuenta las capacidades necesarias para afrontar los desafíos que conllevan los proyectos DGS, lo que implica que futuros ingenieros del software no posean dichas capacidades [31]. Por otro lado, cabe destacar que este modelo de desarrollo está siendo cada vez más utilizado en la práctica y es posible que, en un futuro, se termine convirtiendo en un estándar, por lo que es necesario entrenar a nuestros estudiantes de ingeniería de software para afrontar estas dificultades, ya que se terminarán convirtiendo en los futuros ingenieros de DGS [5].

La gestión y administración es el pilar principal sobre el que gira un proyecto, y en especial un proyecto DGS, ya que es necesaria la organización de un gran número de trabajadores y equipos de desarrollo, a los que se le añade la problemática de gestionar diferentes factores a tener en cuenta como la separación geográfica, la cultura de los diferentes países involucrados en el proyecto o el horario de trabajo en cada país, es por esto que gestionar este tipo de proyectos eficientemente, es un autentico reto. Por lo tanto, es evidente la necesidad de que existan jefes de proyecto altamente cualificados en la gestión de proyectos DGS, para que puedan afrontar la administración del mismo con éxito, solventando todos los impedimentos que puedan ocasionarse. Sin embargo, en la actualidad es complicado encontrar a jefes de proyectos DGS altamente cualificados, con una gran experiencia y con los conocimientos y habilidades necesarias para afrontar correctamente su trabajo. Como consecuencia, son muchas las organizaciones y artículos que se han quejado de la carencia de habilidades y experiencia en los jefes de proyecto DGS, y por lo tanto, la consideran la principal causa del elevado fracaso de estos proyectos [27].

Por consiguiente, es notoria la necesidad de que existan programas de educación que enseñen a nuestro futuros ingenieros de software conocimientos sobre DGS en general, y habilidades (tanto técnicas como no técnicas) necesarias para afrontar con éxito la gestión de este tipo de proyectos en particular. En contraposición, llevar a cabo el entrenamiento y enseñanza de estas habilidades y conocimientos no es una tarea sencilla, puesto que se precisaría la necesidad de introducir a ingenieros de software inexpertos en proyectos reales (para que adquieran esa experiencia necesaria) y por consiguiente las compañías no estén dispuestas a invertir sus recursos en este tipo de programas de entrenamiento. Esta posición de las compañías es debido a que se pueden poner en riesgo proyectos en curso, además de resultar complejo reproducir un escenario real en un entorno de educación [31].

En cualquier caso, hay diferentes formas de llevar a cabo la educación de diferentes conocimientos prácticos y el entrenamiento de ciertas habilidades, sin que esto pueda afectar, en nuestro caso, a un proyecto real. En el campo de la *Educación en la Ingeniería de Software* se han realizado avances, buscando la manera más efectiva de educar ciertos conocimientos a estudiantes de ingeniería de software, apareciendo métodos tradicionales como proyectos finales, combinación de diferentes técnicas de enseñanza como aprendizaje basado en proyectos [3], innovadoras estrategias como las clases volteadas [13], o darle un enfoque relacionado con el uso de juegos, apareciendo el termino de *Gamificación* [17].

Dentro de la gamificación podemos encontrar diferentes tendencias como pueden ser: los cursos académicos [32], los entornos de aprendizaje [9] o las aplicaciones que presentan escenarios reales, conocidos como *Juegos Serios* (JSs) [28]. En especial, los JSs (también llamados juegos educativos), según [11], çonsisten en juegos que van más allá del puro entretenimiento y constituyen una potente herramienta que permite a sus jugadores experimentar y aprender de sus errores, adquiriendo así experiencia y conocimientos". Los JSs ayudarán en el proceso de aprendizaje mediante la simulación de entornos virtuales, sin el riesgo que conllevaría tener al estudiante en un entorno real [5, 11, 27], además de que la tendencia del desarrollo de JSs ha tenido una gran aceptación en la última década.

Como resultado de lo cual, este *Trabajo Fin de Carrera* (TFG) se centrará en el desarrollo de un JS, llamado *GLOBAL-MANAGER*. El objetivo de GLOBAL-MANAGER será el de ayudar a estudiante en ingeniería de software a adquirir y entrenar ciertas habilidades (en especial aquellas no técnicas también llamadas *soft skills*) necesarias cuando se aborda el papel de jefe de proyecto DGS. El jugador tendrá que abordar la gestión de un proyecto DGS ficticio desde el comienzo hasta la entrega del producto software al cliente, tratando de resolver los diferentes impedimentos que se puedan ocasionar en el ciclo de vida. De esta manera, los jugadores podrán adquirir experiencia de una manera sencilla, barata e independiente, permitiéndoles afrontar con éxito un futuro trabajo en la gestión de un proyecto DGS. En la figura 1.3, se muestra un diagrama del funcionamiento que se permite conseguir con el desarrollo de la propuesta anteriormente citada.

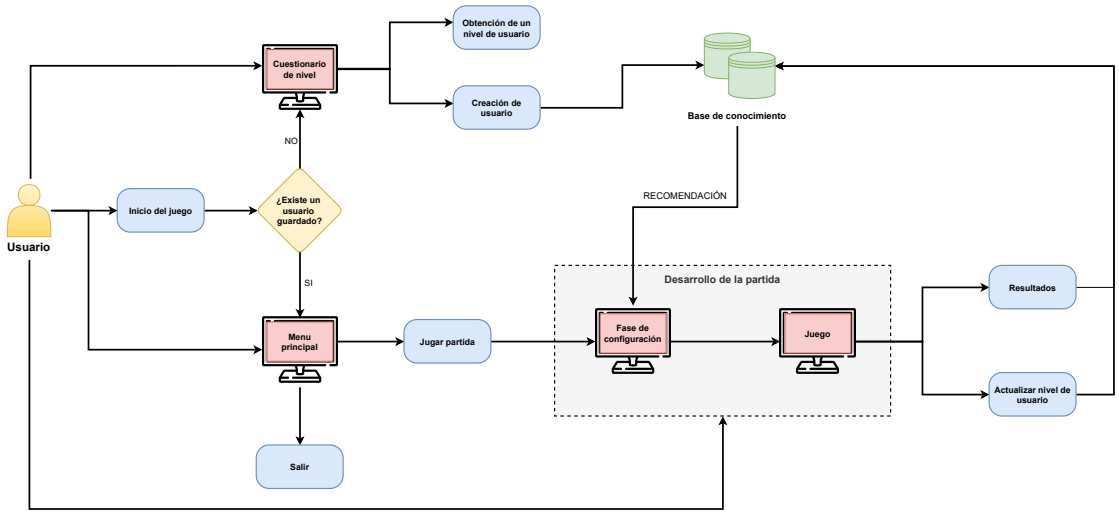

Figura 1.3: Diagrama del funcionamiento del JS propuesto, GLOBAL-MANAGER

Este TFG se enmarca dentro de un contrato I+D con el grupo de investigación Alarcos[1] de la Universidad de Castilla La-Mancha (UCLM)[2], en concreto en el proyecto "G3SOFT: Ingeniería de Modelos para el Gobierno y Gestión del Desarrollo Global de Software"[3], el cual se centra en la mejora del gobierno y la gestión de proyectos de DGS. En la tab. 1.1 se muestra un resumen de dicho proyecto.

Nombre:	G3SOFT:Ingeniería de Modelos para el Gobierno y Gestión del Desarrollo Global de Software
Financiación:	JJCM Consejería de Educación y Cultura y Deportes, y Fondos FEDER
Referencia:	SBPLY/17/180501/000150
Dirección WEB:	https://alarcos.esi.uclm.es/proyectos/G3SOFT/index.php
Grupo de investigación:	Grupo Alarcos
Universidad colaboradora:	Universidad de Castilla-La Mancha (UCLM)
Investigadores principales:	Francisco Ruiz Gónzalez Aurora Vizcaíno Barceló

Tabla 1.1: Características del proyecto G3SOFT

[1] https://alarcos.esi.uclm.es
[2] https://www.uclm.es/
[3] https://alarcos.esi.uclm.es/proyectos/G3SOFT/index.php

1.3. ESTRUCTURA DEL DOCUMENTO

A continuación, se define la estructura del documento, la cual hace referencia a la memoria del TFG y esta dividida en los siguientes capítulos:

- **Capítulo 1. Introducción:** breve indicativo sobre la motivación, contexto y problemática que engloba este TFG, al igual que la solución que se propone.
- **Capítulo 2. Objetivos:** listado tanto del objetivo principal como los objetivos secundarios que se persiguen con la realización de dicho TFG, además de las tareas necesarias para llevarlo a cabo.
- **Capítulo 3. Estado del arte:** información referente sobre el DGS, la gestión y administración de proyectos DGS, de igual modo se especifican un conjunto de habilidades, las cuales son necesarias para llevar a cabo el trabajo de jefe de proyecto software en un entorno distribuido. Por último, se define e indica la importancia de la gamificación en general, y de los JS en particular, a su vez de una serie de ejemplos sobre JS relacionados con el tema de este TFG.
- **Capítulo 4. Método de trabajo:** especificación de la metodología de trabajo que se seguirá para el desarrollo de este TFG, al igual que las herramientas y tecnologías (tanto software como hardware) que se utilizarán en dicho periodo.
- **Capítulo 5. Resultados:** informe de los resultados obtenidos tras llevar a cabo cada uno de los objetivos y tareas definidas en el capítulo 2, utilizando el método de trabajo definido en el capítulo 4, al igual que todos los posibles problemas e impedimentos que puedan haber surgido en la realización de este TFG.
- **Capítulo 6. Conclusiones:** exposición de la conclusión y trabajos futuros tras haber realizado el presente TFG, del mismo modo que las lecciones aprendidas y una valoración personal.
- **Bibliografía:** sinopsis de las fuentes de información consultadas para la realización de este TFG.

CAPÍTULO 2

OBJETIVOS

Este capítulo se centrará en presentar y explicar de manera detallada cual es el objetivo principal que se persigue con la realización del presente TFG, al igual que los objetivos específicos, donde se especificarán aquellos objetivos funcionales y técnicos necesarios para la elaboración del proyecto. A modo de resumen, al final del presente capítulo en la Tabla 2.1, se mostrarán todos los objetivos que se han definido en el inicio del proyecto.

2.1. OBJETIVO PRINCIPAL

El principal objetivo (OP) del presente TFG consiste en diseñar y desarrollar una aplicación software de escritorio, la cual consistirá en un JS en *2.5D* (*pseudo-3D*[1], la tridimensionalidad de un videojuego en 3D se limita a un plano de dos dimensiones), el cual tendrá como objetivo ayudar a estudiantes y profesionales de la ingeniería de software a adquirir ciertas *soft skills*, las cuales son necesarias para llevar a cabo una correcta gestión de un proyecto DGS.

2.2. REQUISITOS ESPECÍFICOS FUNCIONALES

El actual proyecto está compuesto por una serie de requisitos específicos funcionales (REFs), los cuales han de tenerse en cuenta en el desarrollo del JS *Global-Manager*. Al final del TFG y en especial en el Capítulo 6, se llevará a cabo un análisis para comprobar si dichos requisitos han sido debidamente cumplimentados. Estos REFs son los siguientes:

- **REF1.** El JS diferenciará un total de tres niveles distintos de jugador. Estos niveles se calcularán utilizando técnicas de *inteligencia artificial* (IA), en función de los conocimientos del jugador en la gestión de proyectos y en el modelo de desarrollo de software DGS. Los niveles de usuarios que dispondrá el juego serán:
  - Nivel básico: Se corresponderá a aquellos jugadores que posean bajos o nulos conocimientos en la gestión de proyectos o en el DGS, además de no tener una gran experiencia en el desarrollo de software. Son aquellas personas que necesitarán un aprendizaje mucho más lento y prolongado para adquirir todos los conocimientos necesarios para la gestión de proyectos DGS.
  - Nivel intermedio: Se corresponderá a aquellos jugadores que posean ciertos conocimientos en la gestión de proyectos o en el DGS, o incluso poseer cierta experiencia en dichos campos. Estas personas necesitarán de un aprendizaje mucho más rápido, debido a que ya conocerán ciertos aspectos de la materia y les costará menos acostumbrarse al juego.
  - Nivel avanzado: Se corresponderá a aquellos jugadores que posean grandes conocimientos en la gestión de proyectos o en el DGS, además de haber trabajado en numerosas

[1] https://es.wikipedia.org/wiki/2.5D

ocasiones en alguno de estos campos. En este último nivel, los jugadores no necesitarán tanto adquirir conocimientos, sino reforzarlos y entrenarse para mejorar su capacidad de gestionar proyectos DGS satisfactoriamente.

- **REF2.** El JS dispondrá de un *modelo de estudiante* dinámico, es decir, se creará a cada jugador un proceso de aprendizaje en función del nivel de jugador que se le haya definido al principio. Por lo tanto, las partidas que juegue dicho jugador se crearán automáticamente personalizadas a sus conocimientos.
- **REF3.** Las partidas del juego deberán estar divididas en dos fases. La primera fase consistirá en la configuración inicial del proyecto DGS ficticio, la cual estará compuesta por una interfaz gráfica donde se mostrarán un conjunto de parámetros, los cuales son necesarios conocerlos y configurarlos en un proyecto de estas características. El jugador tendrá que fijar cada uno de los parámetros. Además, en tiempo real, se calcularán un conjunto de factores de éxito del proyecto con la configuración actual y un nivel de dificultad del proyecto (el cual podrá tener los siguientes valores *muy fácil*, *fácil*, *normal*, *difícil*, *muy difícil*), que ayudará a saber si la configuración definida es correcta.
- **REF4.** La segunda fase consistirá en una simulación de un proyecto DGS, en donde el jugador adquirirá el rol de jefe de proyecto y deberá administrar dicho proyecto. El jugador tendrá que hacer frente y en algunas ocasiones solucionar diferentes situaciones, a las cuales llamaremos eventos, que puedan ocurrir en el ciclo de vida de un proyecto DGS real. Estos eventos estarán relacionados con las tres ces (los tres grandes desafíos de un proyecto DGS): comunicación, coordinación y control; y podrán ser tanto eventos positivos (repercusión favorable hacia el proyecto), como eventos negativos (repercusión desfavorable hacia el proyecto).

2.3. REQUISITOS ESPECÍFICOS NO FUNCIONALES

Una vez definidos cuales serán los REFs del proyecto en la Sección 2.2, es necesario definir otro tipo de requisitos específicos muchos más técnicos (RENFs). Estos RENFs son los siguientes:

- **RENF1.** En el cálculo del nivel del jugador (anteriormente descrito) se deberán utilizar técnicas de IA, en especial utilizando *Lógica Borrosa*[2]. Al principio del juego y antes de que el jugador juegue una partida, deberá rellenar un formulario con diferentes preguntas sobre sus conocimientos y experiencia en gestión de proyectos y DGS. A través de estas preguntas se obtendrá un nivel para el nuevo jugador. Para implementar esta encuesta y el cálculo automático del nivel se llevará a cabo una entrevista a un experto en la materia, para adquirir aquellos conocimientos que nos hagan saber cuales son los aspectos importantes para conocer las nociones sobre gestión de proyectos y DGS de una persona.
- **RENF2.** Desarrollar el JS utilizando el marco de trabajo de Microsoft *.NET*[3]. Además, se utilizará el lenguaje de programación *C#*[4], el cual se encuentra más enfocado en la programación de videojuegos.
- **RENF3.** La gestión de los datos asociados al proyecto (características de los eventos, modelo del estudiante y dominio) se llevará a cabo utilizando el sistema de gestión de bases de datos relacional *SQLite*[5], junto con el componente *LINQ*[6] (Language Integrated Query) de Microsoft .NET para las consultas a datos de manera nativa a los lenguajes .NET, como C#.

[2]La lógica borrosa, también conocida como lógica difusa, consiste en un enfoque computacional basado en grupos de pertenencia(parcialmente verdadero o parcialmente falso), en vez de en la tradicional lógica booleana de verdadero o falso. https://www.wikiversus.com/informatica/logica-difusa/

[3]https://dotnet.microsoft.com/

[4]https://es.wikipedia.org/wiki/C_Sharp

[5]https://sqlite.org/index.html

[6]https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries

Código Objetivo	*Descripción*
OP	Diseño y desarrollo de un JS en 2.5D, que ayude a sus jugadores a adquirir ciertas soft skills y a entrenarse en la correcta gestión de proyectos DGS.
REF1	Diferenciación de tres niveles de jugador (bajo, medio y alto) para crear asi diferentes procesos de aprendizaje, en función de los conocimientos del jugador en la mataria.
REF2	Creación de un modelo de estudiante dinámico para la creación de partidas personalizadas.
REF3	Primera fase del juego: Implementación de una interfaz grafica, en donde el jugador deberá configurar un conjunto de parametros iniciales del proyecto DGS. Además del calculo, en tiempo real, de un conjunto de fáctores de éxito y el nivel de dificultad del proyecto.
REF4	Segunda fase del juego: Implementación de la simulación del proyecto en función de la configuración inicial definida, el jugador deberá gestionar dicho proyecto DGS haciendo frente a diferentes eventos (positivos y negativos) relacionados con la comunicación, coordinación y control del proyecto.
RENF1	Cálculo del nivel del jugador mediante un formulario, el cual se implementará utilizandológica borrosa, tras realizar una adquisición de conocimientos mediante una entrevista a un experto sobre cuales son los aspectos más importantes para conocer las nociones sobre gestión de proyectos y DGS de una persona.
RENF2	Desarrollo del JS utilizando Microsoft .NET, junto con el lenguaje de programación C#.
RENF3	Gestión de los datos asociados al proyecto mediante el sistema de gestión de bases de datos relacionales SQLite, junto con el componente LINQ de Microsoft .NET.

Tabla 2.1: Objetivos principal y específicos del presente TFG

CAPÍTULO 3

ESTADO DEL ARTE

El objetivo de este capítulo consistirá en la definición y explicación detallada del **conocimiento relevante a la temática** del presente TFG. A continuación, se presenta el estado del arte del **paradigma** de desarrollo DGS, donde se detallará su definición, beneficios, problemáticas. Además, se indicará la importancia que conlleva la gestión en un proyecto de estas características. Por otra parte, se **hará** referencia a las habilidades que son necesarias para llevar a cabo el trabajo de jefe de proyecto en un entorno DGS. A continuación, se tratará la situación actual de la gamificación para la enseñanza, junto con los JS y el uso de la IA para la personalización de juegos. Para finalizar, se listarán una serie de ejemplos relacionados con el objetivo de este TFG.

3.1. DESARROLLO GLOBAL DEL SOFTWARE

Debido a la globalización, en el campo de la ingeniería de software aparece un nuevo modelo de desarrollo denominado como Desarrollo Global de Software. A diferencia del modelo tradicional, donde el equipo de trabajo estaba centralizado en un solo edificio o en varios edificios, pero siempre en el mismo país, el DGS consiste en el desarrollo de un producto o servicio software, en donde diferentes equipos de desarrollo, **pertenecientes o no a organizaciones diferentes, se ubican en distintos países. Estos grupos de desarrolladores colaboran en el desarrollo del** mismo proyecto software, y son coordinados y gestionados en tiempo real basándose en los desafíos del DGS, las 3Cs: comunicación, coordinación y control [38].

A lo largo de la última década, el DGS se ha afianzado como una de las vertientes más relevantes en la investigación dentro del campo de la ingeniería del software. Las primeras prácticas de este nuevo tipo de proceso de desarrollo de software surgen hace más de 30 años, con los primeros usos de *outsourcing* [6]. Sin embargo, no sería hasta 2006 cuando se extendiera su uso con la celebración de la primera conferencia internacional sobre DGS, el *ICGSE* (*IEEE International Conference on Global Software Engineering*) [38, 44].

3.1.1. Beneficios del Desarrollo Global del Software

Cuando hablamos de DGS es notable resaltar la cantidad de beneficios que pueden alcanzar las empresas con su correcto uso. Sin embargo, no todos sus beneficios son conocidos en el sector, ya que podemos encontrar beneficios que se puedan obtener de forma directa o indirecta, y son muchas las investigaciones, como [2, 15, 16, 44], que están estudiado tanto aquellos beneficios notables como aquellos no vistos a simple vista con el uso de DGS como proceso de desarrollo de software. Por lo tanto, a continuación se formulará un listado de todos los beneficios que podemos obtener con esta nueva tendencia:

- **Reducción del coste de producción.** Uno de los beneficios más importantes y una de las razones por las cuales las organizaciones y compañías están optando en afrontar un modelo de desarrollo de software global, consiste en la reducción de los costes del proyecto en el proceso

de producción. Este beneficio se debe en gran medida a la globalización, ya que ha hecho posible que actividades en el proceso de desarrollo puedan realizarlas empleados que se encuentran en países que cuentan con salarios más reducidos [2]. Un ejemplo de esta situación sería la India, en donde el salario base anual de un desarrollador de software es de *US$15,000*, mientras que en Irlanda un mismo trabajador puede ganar cuatro veces esa cantidad, y a su vez esa cantidad consistiría en la mitad del salario de un desarrollador en Estados Unidos [15, 16]. Alcanzar esta situación también ha sido posible gracias al despliegue de enlaces de comunicación de alta velocidad, los cuales ayudan a la transferencia de información y conocimientos entre los empleados separados geográficamente [2]. Este beneficio conlleva que las compañías tengan que tener en cuenta la gestión del coste **de** los viajes de empleados entre grupos de trabajo, ya que en este modelo de desarrollo los empleados no se conocen y es necesario que exista comunicación *face-to-face* para consolidar la relación entre empleados, creando así un ambiente de confianza [15, 16].

- **Aprovechamiento de la zona horaria. El DGS** hace posible que las organizaciones puedan aprovecharse de la diferencia en las zonas horarias de sus empleados, con el fin de incrementar las horas de trabajo del día y reducir así el tiempo de desarrollo del servicio o producto software [15, 16]. De esta manera se lograrán jornadas de trabajo más extensas en el proyecto, que en uno con un modelo tradicional y por lo tanto una mayor productividad para finalizar dicho proyecto en un menor tiempo [44]. Esta situación es conocida como desarrollo *follow-the-sun*[1] y es considerada un potente beneficio en el DGS. Sin embargo, lograr este escenario en la realidad es complicado, además de poder producirse retrasos en las respuestas en la comunicación de los empleados, debido a horarios de comida o días festivos, lo que puede ocasionar retrasos en el proyecto. Por lo tanto, es importante que exista cierto solapamiento en las jornadas laborales de los empleados, con el fin de que exista cierta comunicación sincrona [15, 16].
- **Modularización del proceso de desarrollo.** Una manera de afrontar un proyecto DGS es separando las tareas del proyecto en módulos independientes bien definidos, para que de esta manera cada equipo de desarrollo posea ciertos módulos de trabajo. **De esta manera, las decisiones referentes a cada módulo se toman de forma aislada entre los miembros del equipo de trabajo y se reducen costes en la coordinación del proyecto**. De esta manera, podemos diferenciar dos tipos de estrategias: *basada en módulos*[2] y *basada en fases*[3] [2, 15, 16].
- **Acceso a plantillas de trabajadores altamente cualificados.** El DGS hace posible acceder fácilmente a grupos de trabajadores altamente cualificados repartidos por todo el mundo. De esta manera, las empresas se pueden beneficiar de los conocimientos, diversidad de experiencias, destrezas y habilidades de trabajadores repartidos por todo el mundo, para que lleven a cabo el desarrollo de diferentes actividades software del proyecto DGS [2, 15, 16, 44].
- **Proximidad al cliente y a su mercado.** Gracias al DGS, las compañías pueden establecer fácilmente filiales en aquellos países donde se localizan los clientes, y conseguir así un acercamiento y un conocimiento más profundo del mercado local. Además, con esta practica las compañías consiguen expandirse hacia nuevos mercados, sin la necesidad de trasladar a sus equipos de desarrolladores [2, 15, 16, 44].

[1] "La estrategia follow-the-sun se caracteriza en que cuando un equipo de trabajo finaliza su jornada laboral, la jornada de otro equipo comienza en otra parte del mundo, de esta manera se consigue un desarrollo del proyecto las 24 horas del día"[38]

[2] "La estrategia basada en módulos consiste en dividir el proyecto en diferentes módulos, los cuales se pueden considerar un artefacto completo del proyecto, y repartirlos entre los sites"[38]

[3] "La estrategia basada en fases consiste en asignar a cada equipo de trabajo una fase del proceso de desarrollo software"[38]

3.1.2. Desafíos del Desarrollo Global del Software

La distancia existente entre los equipos de desarrolladores en un proyecto DGS hace posible el acercamiento hacia grandes beneficios por parte de las organizaciones, sin embargo conseguirlos no es tarea sencilla, ya que dicha distancia conlleva también la introducción de un conjunto de desafíos, a los cuales, las organizaciones deben hacer frente para alcanzar los beneficios anteriormente citados [15]. Los desafíos que puedan aparecer en este tipo de proyectos se pueden agrupar en relación con los tres grandes procesos en el desarrollo de software: comunicación, coordinación y control. Estos grupos de desafíos también se les conoce como las 3 Cs [38, 44], y hacen referencia a las siguientes situaciones:

- **Desafíos de comunicación.** Hace referencia a aquellas situaciones en donde se lleva a cabo una comunicación entre trabajadores, es decir, un intercambio de información y conocimientos con el fin de que no se produzcan malentendidos y el proyecto pueda avanzar.
- **Desafíos de coordinación.** **Hace referencia a la dirección y planificación del conjunto de trabajadores junto con las diferentes tareas de un proyecto**, con el fin de alcanzar objetivos e intereses comunes y la evolución del proyecto progrese adecuadamente.
- **Desafíos de control.** Hace referencia a la administración y gestión del proyecto en general, teniendo en cuenta día a día diferentes aspectos del proyecto como pueden ser los calendarios de entregas, presupuestos, calidad, estándares, etc.

Por otro lado, los proyectos DGS se caracterizan por la existencia de diferentes nacionalidades, organizaciones e inclusos culturas que pueden agravar esta situación. Esto hace que puedan aparecer diferentes tipos de distancias entre los miembros del proyecto, provocando una acentuación en los desafíos anteriormente citados [44]. De esta manera, aparece otra clasificación de los desafíos de un proyecto DGS en función de lo que se conoce en la literatura como las tres distancias [15, 16, 44]:

- **Distancia geográfica.** Se puede definir como la medida de esfuerzo necesario en un individuo para visitar un punto alejado de su ubicación. Un ejemplo sería el de dos ubicaciones separadas geográficamente por una gran distancia pero con un enlace aéreo directo frente a dos ubicaciones cercanas geográficamente pero con poca infraestructura de transporte; de esta manera el primer caso poseerá poca distancia geográfica, mientras que en el segundo será elevada [44]. Además, la distancia geográfica dificulta la posibilidad de existir una comunicación informal o cara a cara entre los miembros que ayude a afianzar las relaciones entre ellos, el trabajo en equipo, consolidación de la confianza y la comunicación fluida de información importante del proyecto [15].
- **Distancia temporal.** Ligada a la anterior, se puede definir como la medida de la diferencia en el tiempo existente en la comunicación entre dos individuos [44]. Como se dijo en la sección anterior, la distancia temporal en un proyecto DGS puede conllevar ciertos desafíos como es la estrategia follow-the-sun, con el objetivo de reducir tiempo y costes, sin embargo, surgen también ciertas problemáticas como es el hecho de que aparezcan retrasos en las respuestas en los intercambios de información entre los trabajadores, debido a la reducción del solapamiento de horas en las jornadas laborales de los equipos de desarrolladores. Esto implica que se tengan que usar herramientas de comunicación asíncronas, las cuales pueden afectar negativamente al manejo de ambigüedades y al aumento de los malentendidos, produciéndose así retrasos en el proyecto; frente a una comunicación síncrona mucho más directa y segura [15].
- **Distancia socio-cultural.** Se puede definir como la medida en que un individuo conoce y comprende las costumbres, cultura e idioma de otro individuo, con el objetivo de llevar a cabo una correcta comunicación. Esta situación es debida a que como en un proyecto coexisten diferentes nacionalidades, es frecuente que existan **diferencias culturales** entre sus miembros [44]. Dicha distancia puede conllevar diferentes interpretaciones en una comunicación o situación, lo que puede obstaculizar la comunicación y coordinación del proyecto [15]. Además, puede provocar que aparezcan conflictos y malentendidos entre los miembros del proyecto,

retrasando así la evolución del mismo.

Por lo tanto, es importante tener en cuenta los posibles desafíos que puedan surgir en un proyecto DGS, es por ello que son numerosas las investigaciones sobre estas problemáticas, como es el caso de [34], en donde se lleva a cabo una Revisión Sistemática de la Literatura (RSL) de 101 estudios sobre las dificultades más importantes en un proyecto de estas características, junto con un cuestionario a 41 organizaciones sobre sus prácticas en el mundo real. A continuación, utilizando como referencia el estudio de [34], se mostrará en la Tabla 3.1 las frecuencias de importancia para las dificultades, que se deben afrontar en los proyectos DGS, en 101 estudios (tab. 3, pág. 5) y para 41 organizaciones (tab. 6, pág. 7), junto con una media ponderada de ambos valores.

Desafío DGS	Frequencia de aparición en 101 estudios (%)	Frecuencia de acuerdo con 41 organizaciones (%)	Media (%)
Falta de entendimiento cultural	62	70	**66**
Ausencia de comunicación	54	76	**65**
Falta de la gestión del conocimiento	38	78	**58**
Falta de gestión del tiempo	42	71	**56.5**
Ausencia de coordinación	35	69	**52**
Ausencia de control	27	75	**51**
Actividades de ingeniería de requisitos	28	71	**49.5**
Asignación de tareas	18	80	**49**
Ausencia de la verdad	34	59	**46.5**
Actividades de gestión del cambio	22	68	**45**
Falta en la concienciación de equipo	23	66	**44.5**
Gestión de conflictos	17	71	**44**
Estimación del coste y del esfuerzo	15	73	**44**
Actividades de integración	14	73	**43.5**
Gestión del riesgo	15	71	**43**
Distancia geográfica	28	56	**42**
Ausencia de un proceso uniforme entre los diferentes sites	19	63	**41**
Falta de una infraestructura informatica adecuada	11	68	**39.5**
Protección de la propiedad intelectual	9	64	**36.5**

Tabla 3.1: Frecuencia de la importancia de diferentes dificultades en proyectos DGS

3.1.3. Rol del jefe de proyecto

Puesto que este TFG se va a centrar en el entrenamiento de los jefes de proyecto, es necesario conocer las habilidades que la persona encargada de este rol debe tener. El jefe de un proyecto posee la tarea de administrar y gestionar un proyecto, lo cual consiste en la aplicación de un conjunto de técnicas y herramientas con el objetivo de controlar los recursos de un proyecto y la correcta realización de las diferentes tareas de los miembros, para conseguir así una evolución constante y sin problemas en el desarrollo del servicio o producto software. Dicha tarea esta ligada a diferentes aspectos relevantes de un proyecto, como es la gestión del tiempo, el coste, la calidad y muchos otros. Por lo tanto, el rol de jefe de proyecto es de suma importancia, ya que controlará la consistencia y evolución del proyecto, además de asegurarse de que los objetivos y propósito del mismo se cumplan correctamente y sin incidencias. El jefe de proyecto deberá hacer frente a cualquier contratiempo que pueda surgir a lo largo del proceso de desarrollo del software [14]. Cabe destacar, que en la literatura se considera al proceso de gestionar un proyecto software una tarea esencial para alcanzar el éxito, sin embargo, también se considera dicho proceso como una tarea no sencilla, en la cual se necesita **conocimiento, experiencia y habilidades** en el desarrollo de proyectos, según [7] definió dicho proceso como la integración de la tecnología del software, la economía y las relaciones laborales en el contexto de un proyecto software.

Por otro lado, si hablamos de la gestión de proyectos en un entorno global como es un proyecto DGS, dicha labor se complica aún más. Con la separación de los equipos de desarrolladores de un mismo proyecto en diferentes ubicaciones con sus diferencias temporales, culturales y lingüísticas, implica que el rol de jefe de proyectos DGS sea mucho más complicado, siendo necesaria la organización de trabajadores no conocidos personalmente a diferentes tareas, junto con la gestión de recursos ubicados en distintos lugares del mundo. Adicionalmente, dicha tarea se agrava con los desafíos

en las llamadas 3 Ces, comunicación, coordinación y control, siendo un gran número los aspectos y desafíos que debe afrontar un jefe de proyectos DGS con el objetivo de alcanzar exitosamente la correcta colaboración entre los equipos de desarrolladores y la entrega del servicio o producto software al cliente. Debido a esta problemática, es **necesario enseñar** a los estudiantes de ingeniería de software y a los futuros jefes de proyectos DGS los desafíos de este nuevo paradigma de desarrollo de software como en el proceso más importante en este tipo de proyectos**, la gestión del mismo**. Por lo que, será necesario un entrenamiento y enseñanza de aquellas habilidades, no solo técnicas, sino en especial aquellas no-técnicas (*soft skills*) necesarias para afrontar con éxito los posibles desafíos que puedan surgir en un proyecto DGS.

3.2. HABILIDADES NECESARIAS EN DESARROLLO GLOBAL DEL SOFTWARE

Como hemos dicho anteriormente, trabajar en un entorno distribuido como es un proyecto DGS, resulta complicado, ya que aparecen nuevas situaciones problemáticas debido a las dificultades en la comunicación y a la separación geográfica, temporal y socio-cultural. Además, en el proceso de gestión del proyecto esta situación se agrava siendo necesarios más conocimientos para cumplir correctamente con su trabajo. Por lo tanto, son necesarias ciertas habilidades para trabajar en **este tipo de proyectos**. Aunque, ambas, habilidades técnicas y no-técnicas son igual de importantes en un proyecto software, cabe destacar que las habilidades no-técnicas resultan ser más difíciles a la hora de enseñarlas y/o aprenderlas. Por lo tanto, en esta sección nos centraremos en listar y explicar cuáles son las habilidades no-técnicas más importantes y necesarias cuando se trabaja en un entorno de desarrollo distribuido y cuando se lleva a cabo la labor de jefe de proyecto.

3.2.1. Habilidades en el equipo de trabajo de Desarrollo Global del Software

Al igual que se citó anteriormente, el elevado nivel de fracaso en los proyectos DGS es debido a la falta de habilidades y competencias relacionadas con este nuevo modelo de desarrollo entre sus trabajadores. Es por esto, que son numerables los estudios que intentan recopilar el conjunto de habilidades necesarias para trabajar en un entorno DGS. Algunos de estos estudios son los siguientes:

En primer lugar, según [8], algunas de las habilidades más útiles en el desarrollo de software son *hablar un idioma extranjero (en especial el inglés)*, *capacidad de tener una cooperación local* y *ser capaz de tomar decisiones*. En adicción a las anteriores, en un entorno distribuido se hacen necesarias nuevas habilidades como *ser capaz de mantener una cooperación remota*, *llevar a cabo una cooperación intercultural* y *una cooperación con el cliente*.

Por otra parte, y de acuerdo con [29], algunas de las habilidades que se deben promover en el entrenamiento de DGS son *ser conscientes de todos los problemas posibles*, *dominar protocolos para la correcta comunicación entre trabajadores (en especial con el uso de ordenadores)*, *correcta comunicación oral y escrita a través de un idioma común*, *conocer los códigos de ética de la organización* y *llevar a cabo una correcta gestión del tiempo*.

En [37], los autores indican que las habilidades más importantes que deben enseñarse para aprender a trabajar en DGS son *comunicación regular entre los miembros de los equipos de desarrolladores distribuidos*, *contribuir a la dinámica del equipo de trabajo*, *saber cómo trabajar en equipos culturalmente divergentes*, *gestión del tiempo* y *saber trabajar con tecnologías de colaboración*.

Igualmente, un marco de trabajo para enseñar algunas habilidades de DGS se describe en [18]; las principales habilidades consideradas son *saber comunicarse mediante ordenadores*, *desarrollo iterativo en las relaciones cliente-desarrollador a distancia* y *gestión distribuida de proyectos*.

Por último, el modelo de competencias especificado en [40] indica que las habilidades necesarias en un entorno DGS se pueden dividir en cuatro grupos, dependiendo del rol que se posee en el proyecto,

estos roles pueden ser *ingeniero de software*, líder de equipo, *gestor de proyecto* y gerente de la unidad organizadora. Las habilidades que se señalan como necesarias en el entrenamiento de ingenieros de software DGS son *comunicación síncrona y asíncrona*, *identificación y gestión de las necesidades del proyecto*, *capacidad de solventar problemas técnicos*, *conocimiento de técnicas avanzadas de comunicación distribuida*, *capacidad de autoaprendizaje*, *capacidad de mantener relaciones internacionales*, *uso de tecnologías de la comunicación y la información*, *capacidad de trabajar en un entorno global* y *saber mantener una comunicación oral y escrita en inglés*.

Teniendo en cuenta todas las habilidades y competencias citadas anteriormente encontradas en la literatura, se muestra en la Tabla 3.2 un resumen de aquellas soft skills más importantes para aprender a la hora de trabajar en un entorno DGS.

Soft Skills	Descripción de la soft skill	Referencias
Comunicación oral y escrita en Inglés	Saber comunicarse oralmente y por escrito con los diferentes miembros de los equipos de trabajo en un idioma común, en especial el Inglés	[8, 29, 40]
Comunicación regular entre los miembros de los equipos de desarrolladores distribuidos	Mantener una comunicación fluida con los diferentes miembros de los equipos distribuidos, sabiendo en cada caso cuál es el mejor protocolo a utilizar (síncrono o asíncrono) y utilizando técnicas avanzadas	[18, 29, 37, 40]
Toma de decisiones y resolución de problemas	Ser capaz de tomar las decisiones adecuadas para resolver los problemas técnicos que afectan directa o indirectamente a la evolución del proyecto, considerando todas las problemáticas posibles	[8, 29, 40]
Gestión del tiempo	Gestionar el tiempo empleado en el desarrollo de las tareas de los diferentes equipos de trabajo, en un esfuerzo por evitar problemas en el proyecto	[18, 29, 37]
Cooperación remota	Colaborar tanto entre los miembros del mismo equipo como entre los diferentes equipos de trabajo para dinamizar el proyecto, utilizando también herramientas y tecnologías que ayuden en la cooperación	[8, 18, 37, 40]
Cooperación intercultural	Saber cómo trabajar y cooperar con personas de diferentes culturas y tener la capacidad de relacionarse internacionalmente	[8, 37, 40]
Cooperación con el cliente	Cooperar con los clientes que nos han contratado para llevar a cabo el proyecto, identificando y gestionando los requisitos que quieren para tratar de completar el proyecto con éxito	[8, 18, 40]
Gestión del conocimiento	Gestionar toda la información y el conocimiento que se crea en un proyecto distribuido y tratar de que llegue a todos los equipos distribuidos involucrados	[40]
Códigos de ética	Conocer los código de ética de un proyecto distribuido y trabajar en cumplimiento de estos	[29]

Tabla 3.2: Conjunto de las soft skills más importantes para trabajar en proyectos DGS

3.2.2. Habilidades en jefes de proyecto de Desarrollo Global del Software

Una vez se ha llevado a cabo el anterior análisis sobre las habilidades y competencias necesarias para trabajar en un entorno DGS, a continuación nos centraremos en las competencias necesarias para llevar a cabo el rol de jefe de proyecto.

En primer lugar, los autores en [43], indican que el riesgo es una de las partes más importantes en la gestión de un proyecto. Estos es debido a que en un proyecto de software distribuido los jefes de proyecto tienen que tener en cuenta áreas, en donde pueden surgir riesgos como puede ser *la zona horaria*, *la diferencia cultura*, *la distancia geográfica* y *la diferencia lingüística*, al igual que los procesos de *coordinación*, *comunicación* y *control*.

En adicción a las anteriores consideraciones, la investigación en [42] ofrece un conjunto de habilidades las cuales deben de ser adquiridas por el futuro jefe de proyecto, para conseguir así lograr los objetivos de los proyectos que gestione. Las habilidades que se proponen son *habilidades de comunicación*, *habilidades para la construcción de* ***espíritu de equipo*** y *habilidades para la resolución de problemas*.

Por último, las competencias que se proponen en [40] como necesarias en el entrenamiento de jefes de proyectos DGS son *la toma de decisiones*, *la gestión de reuniones*, *el establecimiento de reglas para trabajar en un ambiente con datos compartidos*, *saber recopilar, analizar e interpretar la información*, *poseer una actitud positiva y una capacidad para motivar a los demás*, *capacidad para organizar y planificar*, *poseer iniciativa y liderazgo*, *capacidad de resolver conflictos interpersonales*,

identificación de competencias en Currículum-Vitae (CV), estimación de las necesidades y establecimiento de prioridades.

Después de analizar y estudiar el conjunto de soft skills requeridas para llevar a cabo actividades de gestión de proyectos en entornos distribuidos, a continuación se muestra en la tabla 3.3 una agrupación de aquellas soft skills más importantes para aprender a gestionar un proyecto DGS correctamente.

Soft Skills	Descripción de la soft skill	Referencias
Actitud positiva y capacidad para motivar a otros	Poseer la capacidad de animar a los diferentes miembros que componen el proyecto de software distribuido, aumentando así su capacidad de trabajo	[40]
Iniciativa y liderazgo	Tener la capacidad de dirigir y gestionar el proyecto distribuido, tomando la iniciativa en las ideas	[40]
Toma de decisiones y resolución de problemas	Poder tomar decisiones y así resolver los problemas (tanto técnicos como interpersonales) que afectan directa o indirectamente a la evolución del proyecto, considerando todas las posibles cuestiones involucradas	[40, 42]
Habilidades para la construcción de entornos de equipo	Tener la capacidad de mejorar y fortalecer la relación y la comunicación con otros miembros y equipos de trabajo, creando así un ambiente de trabajo confortable	[42]
Habilidades de comunicación	Poder comunicarse adecuadamente con los demás miembros del proyecto, así como con los clientes del mismo, eligiendo un idioma común en ambos casos y teniendo en cuenta las diferencias culturales, geográficas y de zona horaria	[42, 43]
Habilidades de coordinación	Saber combinar y gestionar todos los detalles técnicos y personales en las diferentes tareas de las que se compone el proyecto distribuido, para completarlo con éxito	[40, 43]
Habilidades de control	Poder examinar la evolución del proyecto distribuido para comprobar si la retroalimentación obtenida se corresponde con los requerimientos de los clientes, recogiendo, analizando e interpretando la información obtenida	[40, 43]
Estimación de las necesidades y establecimiento de prioridades	Saber priorizar y estimar las diferentes tareas del proyecto distribuido en función de las demandas y requerimientos de los clientes en cada momento	[40]
Gestión de reuniones	Tener la capacidad de llevar a cabo la gestión de la reunión, tanto con los diferentes miembros del proyecto distribuido como con los clientes	[40]
Identificación de competencias en CV	Ser capaz de identificar cuáles son las habilidades necesarias para llevar a cabo una tarea, y decidir qué trabajador es el más apropiado para esa tarea, según sus habilidades	[40]
Establecimiento de reglas para trabajar en un ambiente con datos compartidos	Saber imponer las reglas adecuadas en la gestión de los datos compartidos para que lleguen a todos los equipos de trabajo	[40]

Tabla 3.3: Conjunto de las soft skills más importantes para trabajar en proyectos DGS como jefe de proyecto

3.3. GAMIFICACIÓN E INTELIGENCIA ARTIFICIAL

La formación de los estudiantes consiste en un aspecto muy importante a tener en cuenta, ya que implica la profesionalidad con la que los futuros profesionales del sector se desenvolverán en el ámbito laboral. A lo largo de la historia, los expertos han buscado la forma más eficiente y sencilla de transmitir conocimientos a los alumnos. Con la llegada de las nuevas tecnologías al ámbito de la educación, ha provocado que en los últimos años aparezcan nuevas técnicas innovadoras de enseñar diferentes conocimientos y habilidades a los alumnos.

En el campo de la *Educación en la Ingeniería de Software* se han realizado avances con el objetivo de buscar la manera más eficiente de educar a los estudiantes de Ingeniería de Software. Algunos de los métodos que se han utilizado son: uso de métodos tradicionales como proyectos finales, combinación de diferentes técnicas de enseñanza como aprendizaje basado en proyectos [3], innovadoras técnicas como las clases invertidas [13], o la introducción de enfoques relacionados con los juegos, apareciendo el término de Gamificación [17].

El término de Gamificación consiste en aplicar los principios, diseños y elementos propios de un juego/videojuego en cualquier proceso, como la Educación, más allá del contexto de los juegos/videojuegos [23]. El objetivo de la Gamificación consiste en crear experiencias personalizadas, con el fin

de transmitir ciertos sentimientos y una participación por parte del jugador cuando se juega, más allá del mero propósito de entretener [4]. **La Gamificación consiste en una de las técnicas de aprendizaje más utilizadas** en la adaptación de las nuevas tecnologías en el campo de la educación. Dentro del ámbito de la Gamificación podemos encontrar diferentes tendencias como pueden ser: cursos académicos adaptados [32], los entornos de aprendizaje [9], o la creación de aplicaciones que presentan escenarios reales simulados, conocidos como Juegos Serios o Juegos Educativos [28].

Por otro lado, el auge que ha tenido la IA en los últimos años, ha hecho posible que los videojuegos se encuentren más ligados a la IA, utilizándose e introduciéndose diferentes técnicas como sistemas basados en reglas, minería de datos o *machine learning*. Estos métodos se han utilizado para introducir nuevos aspectos en los videojuegos, como son los perfiles de jugador, balances de complejidad o generadores de niveles [46].

3.3.1. Juegos Serios

El término JS apareció por primera vez en 1987, formalizado por el investigador estadounidense Clark Abt en su libro *Serious Games* [1]. Los JS, también conocidos como juegos educativos o de entrenamiento, consisten en herramientas de aprendizaje, cuyo propósito va más allá de la pura diversión, es decir, esta clase de juegos posee un valor educativo y se centra en el propósito académico. Los JS representan potentes herramientas que permiten a sus jugadores entrenar y adquirir ciertas habilidades, gracias a la experimentación, la incitación de la participación, el aprendizaje mediante sus propios errores cometidos, ya que se ofrecen recompensas o castigos inmediatos, adquiriéndose de esta manera cierta experiencia. La principal intención de los JS es que sirvan para un ambiente educativo, pero también posean cierto carácter motivador y atractivo hacia la audiencia objetivo [11, 22, 23]. De esta manera, esta clase de juegos nos permitirán la creación de entornos virtuales, ayudándonos en el proceso de aprendizaje y reduciendo el miedo a cometer errores, sin el riesgo que conllevaría tener al estudiante en un entorno real, ya que muchas compañías no están dispuestas a delegar sus recursos en usuarios inexpertos dentro de programas de entrenamiento [31].

Diversos estudios respaldan que el proceso de jugar favorece el aprendizaje, ya que debido a la diversión se produce una inmersión total del jugador en la tarea que se está realizando, lo que provoca que el aprendizaje y la memorización se vean mejorados. Esta clase de enseñanza se refuerza en el aprendizaje por ensayo-error, debido a que es necesario que durante la evolución del juego, se produzca una retroalimentación inmediata hacia el jugador con información relevante tras cada acción realizada. Además, cabe destacar que dicha clase de juegos posee características diferentes a otras clases de enseñanza, y por lo tanto no deben ser utilizados ni sustituir a la enseñanza principal o tradicional, sino como una herramienta de refuerzo o complemento para fortalecer los conocimientos [23].

Por todo esto, el uso de los JS como medio de aprendizaje se ha convertido en un campo en auge en los últimos años, por lo que resulta conveniente la creación de JS, como herramienta de apoyo, para la enseñanza y entrenamiento de diferentes habilidades de una manera divertida y entretenida.

3.3.2. Inteligencia artificial en Juegos serios

El crecimiento que se ha producido en el campo de la IA, durante los últimos años, ha hecho posible que sean numerosos los algoritmos y técnicas que se han introducido en el diseño de videojuegos, con el objetivo de introducir aspectos y funcionalidades inteligentes, y coordinar la complejidad y dinámicas de estos juegos. Algunos de estos nuevos métodos son por ejemplo mejorar la creación de gráficos realistas, crear escenarios adaptados para los jugadores, generar niveles, balancear la complejidad en función a los conocimientos del jugador o establecer perfiles de usuarios [46]. Al igual que en los videojuegos, también son utilizadas técnicas de IA para el diseño de JS, y de esta manera conseguir JS inteligentes. A continuación, utilizando como base el estudio [22], en donde se

lleva a cabo un revisión sistemática de las técnicas de IA más utilizadas para el desarrollo de JS, se nombrarán y explicarán dichas técnicas junto con las funcionalidades que proporcionan.

La rama de la IA, *Toma de Decisiones* (o en inglés *Decision-Making*), se corresponde a un conjunto de técnicas y algoritmos utilizados para que el sistema en donde se integran sea capaz de tomar decisiones lógicas y racionales a través de la información que se recopila. La entrada de estos algoritmos consistirá en toda la información y conocimiento previo del sistema, la cual dará como salida una determinada acción. Algunos de estas técnicas son:

- **Árboles de decisión**[4] **(Decision Tree).** Los árboles de decisión consisten en la creación de modelos de predicción compuestos por un conjunto de reglas de decisión, los cuales se ejecutan con los datos que se obtienen del sistema durante su funcionamiento. Esta técnica suele ser implementada para modelar el flujo del juego con el objetivo de proporcionar experiencias personalizadas en función de las interacciones del jugador o para evaluar la motivación y comportamiento del mismo.
- **Lógica difusa**[5] **(Fuzzy Logic).** Esta técnica consiste en un enfoque computacional basado en el razonamiento aproximado y los grupos de pertenencia, para ir más allá de la lógica tradicional booleana de verdadero o falso. En el campo de los JS suele utilizarse principalmente para adaptar la jugabilidad en función del jugador, evaluar y clasificar a los usuarios, para crear experiencias inmersivas y controlar *personajes no jugables* (Non-player character, NPC), o incluso para enseñar conocimientos de lógica borrosa.
- **Sistemas de Márkov**[6] **(Márkov Systems).** Su comportamiento va más allá de la lógica difusa y evalua el significado de los valores de la verdad o de la pertenencia, son representadas como un conjunto de transiciones finitas entre un número determinado de posibles estados. Sus usos pueden ir desde la adaptabilidad del juego hasta la recopilación de información del jugador para evaluarlo.
- **Comportamiento orientado a objetivos (Goal-oriented behavior).** El comportamiento orientado a objetivos consiste en un conjunto de técnicas que producen una secuencia de acciones para lograr un determinado objetivo. Está compuesto por un conjunto de objetivos y acciones. Se suelen utilizar para controlar el comportamiento de los NPCs, modelar escenarios o evaluar las respuestas del jugador.
- **Sistemas basados en reglas**[7] **(Rule-Based Systems).** Consisten en sistemas que almacenar y manejan conocimientos en forma de reglas, con el objetivo de interpretar información de una forma rápida y eficiente. Esta técnica es utilizada especialmente para evaluar y guiar a los jugadores, modelar comportamientos para NPCs o crear experiencias de usuario personalizadas.
- **Máquina de estados finitos**[8] **(Finite-State Machines).** Consisten en un modelo matemático computacional, en donde se especifican un conjunto de eventos o condiciones que se deberán cumplir para que el sistema cambie entre diferentes estados, por lo que una salida dependerá no solo de la entrada actual, sino también de las anteriores. Este modelo es utilizado para llevar a cabo el control del flujo del juego (como cambio de niveles o personalización del juego), para evaluar el comportamiento del usuario o controlar NPCs.

Por otro lado, dentro del campo de la IA podemos encontrar la rama *Aprendizaje Automático*[9] (o en inglés *Machine Learning*), el cual se centra en la creación de técnicas que permitan a los sistemas aprender y poseer cierta experiencia, es decir que sean capaces de mejorar y optimizar la realización de una determinada tarea. Diferentes técnicas de esta rama se han utilizado para introducir funcionalidades en JS, algunas de estas son:

[4] https://es.wikipedia.org/wiki/Árbol_de_decisión
[5] https://es.wikipedia.org/wiki/Lógica_difusa
[6] https://es.wikipedia.org/wiki/Cadena_de_Márkov
[7] https://es.wikipedia.org/wiki/Sistema_basado_en_reglas
[8] https://es.wikipedia.org/wiki/Máquina_de_estados
[9] https://es.wikipedia.org/wiki/Aprendizaje_automático

- **Clasificador bayesiano ingenuo**[10] **(Naive Bayes Classifier).** Este tipo de clasificadores consiste en un clasificador supervisado que genera modelos de predicción basado en el teorema de Bayes[11]. Esta técnica consiste en el almacenamiento de clases junto con su conjunto de atributos en una tabla de contingencia. Algunos de los usos que tiene esta técnica son para analizar y evaluar usuarios en función de sus acciones o adaptar el flujo del juego.
- **Redes neuronales artificiales**[12] **(Artificial Neural Networks).** Consiste en un modelo computacional inspirado en el comportamiento de las neuronas del cerebro humano. Esta técnica se compone de un conjunto de estados (neuronas), que procesan una entrada (valores de unas características) para producir una determinada salida, la conexión entre las neuronas esta determinada por un peso y es la que le da el conocimiento al sistema. Sus usos en el mundo de los JS consisten en alterar el flujo de juego, para evaluar y clasificar jugadores o para controlar el comportamiento de NPCs.
- **Razonamiento basado en casos**[13] **(Case-Based Reasoning).** El razonamiento basado en casos consiste en una técnica basada en la capacidad de resolver determinados problemas utilizando soluciones adoptadas anteriormente. Estos sistemas empiezan con un conjunto de datos de entrenamiento del cual se extrae un conjunto de conocimientos. Esta técnica suele ser utilizada para adaptar el flujo del juego, generar experiencias de usuario personalizadas o para la evaluación de las acciones de los usuarios.
- **Máquinas de vectores de soporte**[14] **(Support Vector Machines).** Consiste en un conjunto de algoritmos supervisados centrados en el análisis de datos y reconocimientos de patrones. Esta técnica se suele utilizar con el objetivo de predecir, evaluar o clasificar comportamientos de los usuarios, o controlar los comportamientos de los NPCs.

Por último, cabe destacar que una de las funcionalidades principales que buscamos con la implementación del JS que se desarrollará en el presente TFG consiste en la *Adaptación y Evaluación* (TwoA), es decir que el JS sea capaz de evaluar las habilidades y conocimientos del jugador, y balancear la dificultad del juego en función de estos conocimientos. Según [46], la TwoA permite la creación de JS dinámicos, cuyo mecanismo consigue aumentar el interés, la motivación y la experiencia de juego del jugador. Esta técnica consigue que siempre se le ofrezcan al jugador desafíos factibles con los que pueda mejorar sus habilidades y adquirir conocimientos, encontrándose en un punto intermedio entre la frustración (realización de tareas muy complejas) y el aburrimiento (realización de tareas muy sencillas). De esta manera, conseguimos mejorar y optimizar la curva de aprendizaje, ya que se evalúan constantemente las habilidades del jugador para ajustar la dificultad de las tareas a su nivel. Los algoritmos utilizados para conseguir esta adaptabilidad en el juego están basados en los sistemas de clasificación Elo [21] (originalmente utilizados para evaluar las habilidades de los jugadores de ajedrez). Otro ejemplo es TrueSkill [24], el cual consiste en un algoritmo para evaluar jugadores en juegos en línea a gran escala. Computerized Adaptative Practice [25], representa otro algoritmo utilizado para evaluar las habilidades de jugadores en JS.

3.4. TRABAJOS RELACIONADOS CON EL TEMA

En la literatura actual, podemos encontrar que se han realizado avances para llevar a cabo tanto la enseñanza de los nuevos conceptos relacionados con el DGS como para educar en una correcta gestión de proyectos, al unificar estos dos conceptos surge nuestro JS. Sin embargo, son muchas las maneras en las que se ha podido enseñar estos conceptos. Una de ellas es mediante el uso de cursos educativos, como [26] donde se enseña el rol de la comunicación en el DGS, [36] donde se enseña la coordinación en los proyectos DGS o [19] un curso donde diferentes universidades participan

[10] https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo
[11] https://es.wikipedia.org/wiki/Teorema_de_Bayes
[12] https://es.wikipedia.org/wiki/Red_neuronal_artificial
[13] https://es.wikipedia.org/wiki/Razonamiento_basado_en_casos
[14] https://es.wikipedia.org/wiki/Máquinas_de_vectores_de_soporte

para que sus alumnos aprendan a trabajar con la metodología Scrum en un proyecto DGS. Además, Damian et al. proponen en [18] un marco de trabajo para enseñar habilidades DGS a graduados en ingeniería del software.

Por otro lado, también se han desarrollado numerosos JS para enseñar dichos conceptos. En primer lugar, algunos de los JS utilizados para educar en DGS son los siguientes. *VENTURE* [30], el cual consiste en un entorno de entrenamiento virtual, en donde los jugadores podrán interactuar con *Agentes Virtuales* de diferentes culturas, adquiriendo así habilidades relacionadas con el trabajo en equipo, la resolución de conflictos o las diferencias socio-culturales. Otro trabajo relacionado es *GSD Sim* [35], el cual permite a los jugadores gestionar proyectos DGS mediante la asignación de diferentes equipos de trabajo a diversas localizaciones alrededor del mundo.

De la misma manera, existen en la literatura algunos ejemplos de JS que enseñan cómo llevar a cabo una correcta gestión de proyectos. *PMG-2D* [27] pretende enseñar algunos de los principales conceptos que un jefe de proyecto debe conocer cuando se lleva a cabo la gestión de un proyecto, el jugador deberá identificar los interesados del proyecto, crear una hoja de ruta, y contratar y despedir empleados, para adquirir habilidades como la toma de decisiones, la toma de riesgos, o la gestión del coste, tiempo y recursos humanos. Los autores en [20] presentan *SESAM*, un simulador que permite a estudiantes gestionar un proyecto, mediante la lectura y escritura de texto o contratando y despidiendo empleados. *SimSE* [33] consiste en otro juego que enseña gestión de proyectos, en el cual el jugador adquirirá el rol de jefe de proyecto, y tendrá que gestionar un equipo de desarrolladores, asignando tareas, contratando y despidiendo empleados y monitorizando el progreso, con el objetivo de completar el proyecto, de esta manera el jugador aprenderá el ciclo de vida de los proyectos software y habilidades de toma de decisiones. Otro JS ejemplo es *ProDec* [10], el cual ayuda a estudiantes a entrenar conceptos de gestión de proyectos, por lo que el jugador tendrá que enfrentarse a problemas que pueden aparecer en el ciclo de vida de los proyectos software y solucionarlos tomando la decisión oportuna. Posteriormente, se desarrolló una nueva versión, *ProDecAdmin* [12], en donde los profesores eran capaces de crear nuevos escenarios de juego. El último ejemplo de JS que enseñe conocimientos sobre gestión de proyectos es *SCRUMIA* [45], el cual se corresponde a un juego educativo que entrena a estudiantes en técnicas ágiles de gestión de proyectos aplicando SCRUM, y recordar conceptos y artefactos relacionados con SCRUM.

CAPÍTULO 4

MÉTODO DE TRABAJO

En el presente capítulo, se especificará tanto la metodología de trabajo como el proceso de desarrollo que se ha escogido para la elaboración del proyecto. Además, se definirán los diferentes roles que poseerán las personas involucradas en el dicho proyecto, al igual que las herramientas tanto software como hardware utilizadas para alcanzar los objetivos del proyecto.

En primer lugar se ha escogido como marco de trabajo la metodología ágil de *Scrum* [1]. Se ha escogido este marco de trabajo debido a su facilidad para adecuarse a un proyecto que debe realizarse en pocos meses como es el TFG, además de ayudarnos a adaptarnos rápidamente a los posibles cambios de necesidades que puedan aparecer a lo largo del desarrollo.

Para guiar el proceso de desarrollo se ha decantado por utilizar un *Modelo Basado en Prototipos*, debido a su facilidad de emparejarlo con la forma de trabajar de Scrum, completando mediante incrementos las diferentes funcionalidades del producto y consiguiendo en cortos periodos de tiempo, avances y prototipos funcionales.

4.1. SCRUM

Como se ha dicho anteriormente, se ha escogido como marco de trabajo para la gestión del proyecto a Scrum. Dicho marco de trabajo consiste en un conjunto de buenas prácticas para trabajar colaborativamente y en equipo, utilizado para llevar a cabo el desarrollo de proyectos software de una manera ágil. Este marco de trabajo esta basado en un desarrollo mediante interacciones, lo que permite englobar al proyecto en un ciclo de vida de desarrollo software iterativo e incremental. Se ha decantado por utilizar Scrum debido a su facilidad para integrarlo en un proyecto TFG que necesita ser desarrollado de forma ágil en pocos meses, además de su flexibilidad con relación a posibles cambios en los requisitos o a nuevas exigencias del cliente, ya que resulta complicado realizar una definición exacta del producto al principio del desarrollo. Incluso, gracias a Scrum se puede llevar a cabo una aproximación de los tiempos de una manera más sencilla y precisa que utilizando otras metodologías [41].

Según la guía de Scrum, existen tres pilares fundamentales que caracterizan a Scrum de otros marcos de trabajo, estos pilares son:

- **Transparencia.** Todos los aspectos relevantes que afecten a los diferentes procesos del proyecto deben ser visibles para todos los involucrados en el resultado, es decir, dichos aspectos deben ser definidos mediante un estándar común. Por lo tanto, es necesario que todos los participantes compartan un lenguaje común para referirse al proceso, y las personas que desarrollen el trabajo y aquellos que acepten el resultado deben tener un entendimiento común para "Terminado".

[1] https://www.scrum.org/

- **Inspección.** Es necesaria una inspección constante tanto de los artefactos como del progreso que se está desarrollando para alcanzar los objetivos, con el fin de identificar y corregir las posibles variaciones no deseadas que puedan aparecer, ya que puede implicar grandes problemas.
- **Adaptabilidad.** Debido a que es posible que se produzca alguna desviación en alguno de los procesos o artefactos del proyecto, es necesario llevar a cabo ajustes en los procesos y materiales con el fin de minimizar dichas desviaciones para que no se conviertan en problemas mayores.

4.1.1. Roles

Un proyecto que utiliza la metodología Scrum está compuesto por un equipo Scrum, el cual lo componen diferentes miembros con distintos roles. Dicho equipo resulta ser autoorganizado, debido a que el equipo elige la mejor forma de llevar a cabo las diferentes tareas, y multifuncional, ya que dentro del equipo se encuentran las competencias necesarias para el desarrollo del proyecto. El establecimiento de los roles es un aspecto importante y deben ser conocidos por todos los componentes del equipo en todo momento. Estos roles se dividen en tres: *Dueño del Producto*, *Equipo de Desarrollo* y *Scrum Master*.

- **Dueño del Producto (Product Owner).** Hace referencia a la persona responsable de las necesidades tanto del proyecto como del futuro producto software. Debe expresar, organizar y ordenar los requisitos del producto software, además de ser el responsable de aceptar los incrementos con cada iteración.
  El dueño del producto del presente proyecto estará compuesto por los dos tutores del mismo TFG, *Francisco Pascual Romero Chicharro* y *Aurora Vizcaíno Barceló*.
- **Equipo de Desarrollo (Development Team).** Compuesto por el grupo de profesionales que llevarán a cabo el desarrollo tanto de los diferentes incrementos como del producto final, transformando así los requisitos y necesidades del dueño del producto en funcionalidades del producto software. El equipo de desarrollo deberá ser autoorganizado, multifuncional y actúa como un todo.
  El equipo de desarrollo estará compuesta por una sola persona, el cual se corresponde con el autor del presente TFG, *Rubén Márquez Villalta*.
- **Scrum Master.** Persona responsable de garantizar que los diferentes miembros del equipo Scrum están trabajando según la teoría, prácticas y reglas de Scrum. El Scrum Master es un líder al servicio del equipo Scrum, el cual ayuda a maximizar la productividad y el valor del producto, además de favorecer en la mejora de las interacciones entre los miembros y en la gestión y solución de posibles imprevistos.
  El responsable de garantizar que se está cumpliendo la metodología ágil Scrum a lo largo del proyecto y por lo tanto poseer el rol de Scrum Master, consistirá también en el autor del TFG, *Rubén Márquez Villalta*.

4.1.2. Componentes de Scrum

En un proyecto Scrum podemos encontrar diferentes componentes o artefactos, los cuales son característicos de este tipo de metodología y que ayudan a organizar y gestionar mejor el proyecto. En la figura **??** podemos encontrar un resumen de los artefactos utilizados en Scrum, estos son:

- **Lista de Producto (Product Backlog).** Consiste en una pila de requisitos necesarios para conseguir el objetivo de desarrollar el producto final. La lista está compuesta por historias de usuario creadas por el dueño del producto, a las cuales además se les impone un valor y una prioridad, para que de esta manera la lista quede ordenada. El dueño del producto es el responsable de diseñar, gestionar y ordenar esta pila con funcionalidades, requisitos, mejoras y correcciones del proyecto, aunque puede delegar dicha tarea en el equipo de desarrollo.

- **Sprint.** En Scrum a cada iteración se le conoce con el término *Sprint*. Cada Sprint hace referencia a un bloque de tiempo (normalmente de una a cuatro semanas), en donde se desarrolla un incremento del producto utilizable y potencialmente despegable. Antes de comenzar cada Sprint se decide que funcionalidades y requisitos de la lista de producto se implementarán a lo largo del mismo.
- **Pila del Sprint (Sprint Backlog).** Se corresponde con el conjunto de funcionalidades y requisitos de la lista de producto que se han elegido antes de comenzar un determinado Sprint, y por lo tanto, a lo largo del Sprint el equipo de desarrollo se encargara de implementar dicha pila de funcionalidades. Además, se incorpora un plan para la entrega del incremento y un objetivo del Sprint.
- **Incremento.** Consiste en la suma de todos los elementos implementados de la pila del Sprint durante el mismo y el valor de todos los incrementos obtenidos en los anteriores Sprints. Los incrementos consistirán en productos en si mismos y tendrán que estar en condiciones de ser desplegados. El dueño del producto será el encargado de aceptar el incremento.

Además de los anteriores artefactos, Scrum se apoya en un conjunto de reuniones que se realizan a lo largo del proyecto y los diferentes Sprints, con el objetivo de llevar a cabo una continua gestión y comunicación entre los diferentes miembros del equipo Scrum. Estas reuniones son las siguientes:

- **Reunión de Planificación de Sprint (Sprint Planning Meeting)**. Esta reunión se lleva cabo antes de comenzar un Sprint. En ella, todo el equipo Scrum debate sobre las tareas que se desarrollaran a lo largo del Sprint y define la pila del Sprint junto con el objetivo del Sprint.
- **Revisión de Sprint (Sprint Review)**. Esta reunión se lleva a cabo al finalizar un Sprint. En ella, el dueño del producto inspecciona el incremento obtenido en el Sprint y decide si se acepta o no. Además, se actualiza la pila de producto si fuese necesario con nuevas funcionalidades o con las que no se hayan llevado a cabo durante el Sprint.
- **Scrum diario (Daily Scrum)**. Consiste en una reunión de poca duración que se lleva a cabo diariamente durante el desarrollo de un Sprint, con el objetivo de que el equipo de desarrollo sincronice sus actividades, para que todos ellos estén al corriente de las tareas que se realizan a diario.
- **Retrospectiva de Sprint (Sprint Retrospective)**. Esta reunión tiene lugar después de llevar a cabo la revisión de Sprint y antes que la reunión de planificación del siguiente Sprint. Gracias a esta reunión el equipo Scrum se puede inspeccionar a si mismo y comprobar que aspectos se han desarrollado correctamente y cuales han de ser mejorados para el próximo Sprint.

4.2. DESARROLLO BASADO EN PROTOTIPOS

El Desarrollo Basado en Prototipos o Modelo de Prototipos consiste en un estilo de modelo de desarrollo evolutivo. El proceso de desarrollar software requiere una evolución en el producto, ya que se demandan pequeños componentes funcionales antes del desarrollo completo del sistema. Este modelo de desarrollo se basa en la creación de modelos o prototipos del sistema (utilizando poco tiempo y recursos) de manera evolutiva y cada vez con la implementación de más o menos funcionalidades, consiguiendo así una retroalimentación constante con el cliente mostrándose el comportamiento de los componentes desarrollados hasta el momento, antes de la creación del sistema final, ahorrándose así tiempo y costes ante posibles cambios por parte del cliente. De darse en un modelo de desarrollo tradicional en cascada sería más complicado debido a sus limitaciones en la realimentación con el cliente entre sus fases.

El desarrollo basado en prototipos se sustenta en la idea de ayudar al cliente a plantear los requisitos, cuando no se posee una idea clara de lo que se desea, o cuando el desarrollador software tiene dudas sobre la solución elaborada. De esta manera, el cliente conseguirá resultados a corto plazo con una retroalimentación de lo que se está desarrollando, con el fin de refinar los requisitos del producto software, configurar posibles mejoras y validar los resultados obtenidos.

4.2.1. Prototipos

El desarrollo basado en prototipos requiere de la construcción constante de prototipos. Un prototipo[2] en el desarrollo software se define como un modelo de comportamiento o representación que posee parte de las características del sistema final y puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar los requerimientos. En el desarrollo basado en prototipos podemos encontrar dos tipos distintos de prototipos: *Prototipos Evolutivos* y *Prototipos Experimentales*

- **Prototipos Evolutivos.** Consisten en prototipos que se desarrollan en función de unos requisitos iniciales y como su nombre indica van evolucionando con la retroalimentación del cliente, el cual refina los requisitos para conseguir evolucionar hacia el sistema final.
- **Prototipos Experimentales.** También conocidos como Prototipos Desechables, consisten en prototipos que se diseñan con el fin de experimentar con ciertas funcionalidades o tecnologías, o para comprobar el rendimiento de un componente software que debe cumplir algún requisito técnico. Este tipo de prototipos suele desechar al finalizarse.

4.2.2. Etapas del modelo de prototipos

En el modelo de desarrollo basado en prototipos podemos diferenciar un conjunto de etapas principales, las cuales se deberán cumplir para conseguir el desarrollo del producto final. Estas etapas consisten en:

- **Recolección de requisitos.** Se lleva a cabo una recopilación de cuales son los requisitos iniciales que demanda el cliente.
- **Modelado.** Se efectúa un análisis y posterior modelado de casos de uso e historias de usuario en función de las especificaciones del cliente.
- **Construcción del Prototipo.** Se implementa un prototipo que cumpla todas o algunas de las especificaciones definidas por el cliente.
- **Desarrollo, entrega y retroalimentación.** El prototipo desarrollado en la etapa anterior se entrega al cliente, el cual deberá comprobar y decidir si cumple con los requisitos acordados, se producirá así una retroalimentación. Se añaden nuevos requisitos a desarrollar y se procede a construir un nuevo prototipo. De esta manera, se encamina el desarrollo hacia la obtención del sistema final.
- **Entrega del desarrollo final.** Una vez el anterior prototipo haya sido aceptado por el cliente y posea todas las funcionalidades y requisitos acordados, se procederá a la entrega del producto final al cliente.

4.3. MARCO TECNOLÓGICO

Por último, en esta sección se detallan aquellas herramientas y tecnologías que se han utilizado con el fin de desarrollar tanto el producto software como la documentación del presente TFG. Estas herramientas se han dividido en software y hardware.

4.3.1. Herramientas Software

Las herramientas software utilizadas las podemos dividir en función de para que ha servido su uso. De esta manera podemos encontrar herramientas de desarrollo, control de versiones, modelado, documentación, gestores de base de datos y sistemas operativos.

Sistemas operativos

[2] https://es.wikipedia.org/wiki/Prototipo

- **Windows 10 x64 Education**[3]**.** Última versión del sistema operativo Microsoft Windows, destinado a su uso en PCs de entornos de formación y estudio.

Entornos de desarrollo y lenguajes de programación.

- **Unity 2018.4.22 (64-bit) LTS**[4]**.** Versión del año 2018 del motor de videojuegos Unity. Unity consiste en una plataforma de desarrollo para la implementación de videojuegos en todo tipo de plataformas. Se escogió dicha versión debido a que se corresponde a una edición especial diseñada para tener un soporte más duradero.
- **Microsoft Visual Studio Community 2019 16.5.4**[5]**.** Entorno de desarrollo integrado desarrollado por Microsoft sobre el que se pueden desarrollar aplicaciones software para diferentes plataformas.
- **.NET Framework 4.7.1** Marco de trabajo desarrollado por Microsoft sobre el que se sustentará y desarrollará el juego serio.
- **C Sharp.** También conocido como C#, es un lenguaje de programación multiparadigma desarrollado por Microsoft como parte de su plataforma .NET. Dicho lenguaje de programación será utilizado en su totalidad para desarrollar el juego serio.

Modelado.

- **Balsamiq Mockup 4**[6]**.** Aplicación web y de escritorio diseñada para ayudar a usuarios a crear bocetos de interfaces de usuario de una forma sencilla. Dicha herramienta software ha sido utilizada para diseñar los bocetos iniciales del juego serio.
- **Draw.io**[7]**.** Aplicación web y de escritorio que permite la creación de diagramas de todo tipo de una manera cómoda. Gracias a esta herramienta se ha diseñado diferentes diagramas incluidos en la memoria.

Documentación.

- **LaTeX.** Sistema de composición de textos y creación de documentos escritos con alta calidad tipográfica formado por macros de TeX.
- **TeXstudio**[8]**.** Entorno de desarrollo integrado de LaTeXde código abierto y multiplataforma, utilizado para crear la memoria del presente TFG.
- **BibTeX.** Herramienta para el manejo de referencias bibliográficas, la cual da formato a las listas de referencias que se utilizan en los documentos LaTeX.
- **EndNote**[9]**.** Software de gestión bibliográfica que utiliza BibTeXcomo formato nativo, y ha sido utilizada para gestionar las referencias bibliográficas de la memoria del TFG.

Gestores de base de datos.

Control de versiones.

- **Git**[10]**.** Software de control de versiones utilizado para llevar el manteamiento de los cambios de la aplicación.
- **GitHub**[11]**.** Plataforma web para alojar proyectos software, la cual utiliza el sistema de control dde versiones Git.

[3] https://www.microsoft.com/es-es/education/products/windows
[4] https://unity3d.com/es/unity/whats-new/2018.4.22f1
[5] https://visualstudio.microsoft.com/es/vs/community/
[6] https://balsamiq.com/wireframes/
[7] https://drawio-app.com/
[8] https://www.texstudio.org/
[9] http://www.jabref.org/
[10] https://git-scm.com/
[11] https://github.com/

4.3.2. Herramientas Hardware

Las herramientas hardware utilizadas a lo largo del desarrollo de este TFG han sido un ordenador portátil, con el cual se ha llevado a cabo la creación tanto del código como de la documentación. El ordenador cuenta con las siguientes características:

- **Marca:** MSI
- **Modelo:** Leopard Pro GP62 6QF
- **Procesador:** Intel Core i7-6700HQ 4 x 2.60 GHz
- **Memoria RAM:** 16 GB
- **Tarjeta Gráfica:** NVIDIA GTX 960M

CAPÍTULO 5

RESULTADOS

En este capítulo se expondrán los resultados obtenidos a lo largo del desarrollo del proyecto, tras aplicar la metodología definida en el Capítulo 4. De esta manera, se detallará en primer lugar una visión general del proyecto, compuesto por la planificación de las versiones, sus objetivos y el contexto en el que se enmarcan. Además, se desarrollará en este capítulo los sprints que se han llevado a cabo, junto con las funcionalidades que se han incluido en cada una de las versiones.

5.1. VISIÓN GLOBAL

Como se indicó anteriormente el OP de este proyecto consiste en el diseño y desarrollo de un JS de escritorio, cuyo aprovechamiento sirva para la enseñanza y el apoyo hacia aquellos usuarios que desean aprender conocimientos sobre el nuevo modelo de desarrollar software llamado DGS y en especial, como se lleva a cabo la labor de gestionar proyectos de estas características, utilizando técnicas de gamificación e inteligencia artificial.

Grupo Objetivo

El JS obtenido como resultado estará destinado en especial a estudiantes de ingeniería de software que desean aprender conocimientos sobre como se lleva a cabo el desarrollo de software en proyectos DGS, además de experimentar como llevar a cabo la gestión de este tipo de proyectos. Por otro lado, también estará dirigido a aquellos profesionales de la ingeniería del software que deseen ampliar sus conocimientos en otros modelos de desarrollar software, en especial con esta nueva tendencia, y así estar entrenados ante situaciones que puedan ocurrir en este tipo de proyectos y poder trabajar en un futuro en un entorno real.

Necesidades

La aparición de este nuevo modelo de desarrollo y la importancia que está tomando en los últimos años en el mundo laboral, hace necesaria la existencia de una buena educación y enseñanza de los aspectos y conceptos que lo engloban tanto en el ámbito estudiantil como en el laboral. Incluso, se añade la manifestación del fracaso de un gran número de proyectos con entornos globales debido, en especial mediada a su desconocimiento por parte de los miembros y a la difícil gestión que conlleva esta clase de proyectos, por lo que se hace necesario un entrenamiento previo.

Producto

Con este proyecto de TFG, se pretende obtener como resultado un JS de escritorio que permita a los jugadores jugar diferentes partidas, en donde el jugador (con el rol de jefe de proyecto) deberá gestionar un proyecto DGS ficticio, en el cual aparecerán múltiples situaciones o contratiempos que puedan ocurrir en un ambiente de estas características, las cuales deberán ser solventadas por el jugador.

Beneficios

Los beneficios que conlleva la utilización de esta herramienta consisten en el aprendizaje de un nuevo modelo de desarrollo de software, el cual está tomando cada vez más importancia y puede ser desconocido por un gran número de personas en la actualidad. Además, permite entrenarse en un escenario virtual conociéndose así las situaciones que puedan darse en un proyecto real de estas características, y poder afrontar con éxito la gestión del mismo.

El desarrollo del proyecto se ha dividido en tres versiones diferentes, las cuales serán definidas a continuación:

- **Versión 1.** Esta primera versión se corresponderá con la redacción y definición de los requisitos iniciales del futuro JS. Además, se diseñaran los bocetos de las diferentes ventanas y vistas que poseerá la aplicación mediante la plataforma Balsamiq Mockup. Por otro lado, se llevará a cabo el estudio y aprendizaje de la herramienta para el desarrollo de videojuegos, Unity.
- **Versión 2.** Esta versión intermedia se centrará en el diseño de las diferentes interfaces gráficas de usuario de las cuales estará compuesta el juego, junto con el desarrollo del sistema para llevar a cabo una partida. Para ello se implementarán aspectos de gamificación y jugabilidad en el juego. Obtendremos como resultado una aplicación que nos permita jugar diferentes partidas al juego.
- **Versión 3.** Versión final de la aplicación la cual se centrará en la introducción de diferentes aspectos inteligentes sobre el juego, así como un sistema para la asignación de niveles y para poder llevar a cabo una enseñanza acorde con los conocimientos previos de cada usuario.

5.2. ARQUITECTURA

A continuación, se detallará la arquitectura tecnológica que se ha diseñado para conseguir el desarrollo de Global-Manager, además, de las librerías adicionales utilizadas.

En la figura 5.1 se puede observar la arquitectura tecnológica del JS.

Figura 5.1: Arquitectura tecnológica de Global-Manager

Dicha arquitectura posee tres capas:

- **Capa de usuario.** Representa el entorno sobre el cual se podrá ejecutar la aplicación. En nuestro caso, el JS consistirá en una aplicación de escritorio, la cual se podrá ejecutar en los sistemas operativos *Windows*, *Linux* y *Mac OS*.
- **Capa de aplicación.** Consiste en la capa que hace referencia al JS, el cual consistirá en un proyecto *.NET Framework*, en el cual se utiliza el motor de videojuegos y físicas, *Unity* junto con diferentes *scripts* escritos en el lenguaje de programación C#.
- **Capa de persistencia.** Esta última capa hace referencia a la base de datos o modelo de datos, donde se almacenará toda la información relevante tanto de la aplicación como de los usuarios. Para ello, se ha optado por utilizar como motor de base de datos *SQLite*, de esta manera existirá una base de datos SQLite en local y se llevará a cabo el intercambio de información mediante sentencias SQL.

CAPÍTULO 6

CONCLUSIONES Y TRABAJO FUTURO

BIBLIOGRAFÍA

[1] Clark C Abt. *Serious games*. University press of America, 1987.

[2] Pär J Ågerfalk y col. «Benefits of global software development: the known and unknown». En: *International Conference on Software Process*. Springer. 2008, págs. 1-9.

[3] Afra A Alabbadi y Rizwan J Qureshi. «The proposed methods to improve teaching of software engineering». En: *International Journal of Modern Education and Computer Science* 8.7 (2016), pág. 13.

[4] Manal M Alhammad y Ana M Moreno. «Gamification in software engineering education: A systematic mapping». En: *Journal of Systems and Software* 141 (2018), págs. 131-150.

[5] Sarah Beecham y col. «How best to teach global software engineering? Educators are divided». En: *IEEE Software* 1 (2017), págs. 16-19.

[6] Barry Boehm. «A view of 20th and 21st century software engineering». En: *Proceedings of the 28th international conference on Software engineering*. 2006, págs. 12-29.

[7] Barry W. Boehm y Rony Ross. «Theory-W software project management principles and examples». En: *IEEE Transactions on Software Engineering* 15.7 (1989), págs. 902-916.

[8] Ivana Bosnić, Igor Čavrak y Mario Žagar. «Assessing the impact of the distributed software development course on the careers of young software engineers». En: *ACM Transactions on Computing Education (TOCE)* 19.2 (2019), págs. 1-27.

[9] Lisa J Burnell, John W Priest y JB Durrett. «Teaching distributed multidisciplinary software development». En: *IEEE software* 19.5 (2002), págs. 86-93.

[10] Alejandro Calderón y Mercedes Ruiz. «ProDec: a serious game for software project management training». En: *Proceedings of the 8th International Conference on Software Engineering Advances, Venice, Italy*. 2013.

[11] Alejandro Calderón, Mercedes Ruiz y Rory V O'Connor. «A multivocal literature review on serious games for software process standards education». En: *Computer Standards & Interfaces* 57 (2018), págs. 36-48.

[12] Alejandro Calderón, Mercedes Ruiz y Rory V O'Connor. «ProDecAdmin: a game scenario design tool for software project management training». En: *European Conference on Software Process Improvement*. Springer. 2017, págs. 241-248.

[13] Eun Man Choi. «Applying inverted classroom to software engineering education». En: *International Journal of e-Education, e-Business, e-Management and e-Learning* 3.2 (2013), pág. 121.

[14] Ricardo Colomo-Palacios y col. «Project managers in global software development teams: a study of the effects on productivity and performance». En: *Software Quality Journal* 22.1 (2014), págs. 3-19.

[15] Eoin Ó Conchúir y col. «Exploring the assumed benefits of global software development». En: *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*. IEEE. 2006, págs. 159-168.

[16] Eoin Ó Conchúir y col. «Global software development: where are the benefits?» En: *Communications of the ACM* 52.8 (2009), págs. 127-131.

[17] Thomas M Connolly, Mark Stansfield y Thomas Hainey. «An application of games-based learning within software engineering». En: *British Journal of Educational Technology* 38.3 (2007), págs. 416-428.

[18] Daniela Damian, Allyson Hadwin y Ban Al-Ani. «Instructional design and assessment strategies for teaching global software development: a framework». En: *Proceedings of the 28th international conference on Software engineering*. 2006, págs. 685-690.

[19] Daniela Damian y col. «Teaching a globally distributed project course using Scrum practices». En: *2012 Second International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD)*. IEEE. 2012, págs. 30-34.

[20] Anke Drappa y Jochen Ludewig. «Simulation in software engineering training». En: *Proceedings of the 22nd international conference on Software engineering*. 2000, págs. 199-208.

[21] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[22] Maite Frutos-Pascual y Begoñya García Zapirain. «Review of the use of AI techniques in serious games: Decision making and machine learning». En: *IEEE Transactions on Computational Intelligence and AI in Games* 9.2 (2015), págs. 133-152.

[23] Francisco J Gallego-Durán y col. «Panorámica: serious games, gamification y mucho más». En: (2014).

[24] Ralf Herbrich, Tom Minka y Thore Graepel. «TrueSkill™: a Bayesian skill rating system». En: *Advances in neural information processing systems*. 2007, págs. 569-576.

[25] Sharon Klinkenberg, Marthe Straatemeier y Han LJ van der Maas. «Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation». En: *Computers & Education* 57.2 (2011), págs. 1813-1824.

[26] Marco Kuhrmann y Jürgen Münch. «Distributed software development with one hand tied behind the back: A course unit to experience the role of communication in gsd». En: *2016 IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW)*. IEEE. 2016, págs. 25-30.

[27] Jose Eduardo Nunes Lino y col. «Project management game 2D (PMG-2D): A serious game to assist software project managers training». En: *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2015, págs. 1-8.

[28] Andrew Meneely y Laurie Williams. «On preparing students for distributed software development with a synchronous, collaborative development platform». En: *Proceedings of the 40th ACM technical symposium on Computer science education*. 2009, págs. 529-533.

[29] Miguel J Monasor, Aurora Vizcaíno y Mario Piattini. «Training Global Software Development Skills through a Simulated Environment.» En: *ICSOFT (2)*. 2010, págs. 271-274.

[30] Miguel J Monasor y col. «Global Software Development Education: A commercial perspective from a case study». En: *2014 IEEE 9th International Conference on Global Software Engineering*. IEEE. 2014, págs. 173-182.

[31] Miguel J Monasor y col. «Preparing students and engineers for global software development: a systematic review». En: *2010 5th IEEE International Conference on Global Software Engineering*. IEEE. 2010, págs. 177-186.

[32] Christian Murphy, Dan Phung y Gail Kaiser. «A distance learning approach to teaching eXtreme programming». En: *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. 2008, págs. 199-203.

[33] Emily Oh Navarro y André van der Hoek. «SIMSE: An Interactive Simulation Game for Software Engineering Education.» En: *CATE*. 2004, págs. 12-17.

[34] Mahmood Niazi y col. «Challenges of project management in global software development: A client-vendor analysis». En: *Information and Software Technology* 80 (2016), págs. 1-19.

[35] John Noll y col. «GSD Sim: A global software development game». En: *2014 IEEE International Conference on Global Software Engineeering Workshops*. IEEE. 2014, págs. 15-20.

[36] Martin Nordio y col. «An experiment on teaching coordination in a globally distributed software engineering class». En: *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. 2014, págs. 109-118.

[37] Maria Paasivaara y col. «Teaching students global software engineering skills using distributed scrum». En: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, págs. 1128-1137.

[38] Mario Piattini Velthuis, Aurora Vizcaíno Barceló y Félix García Rubio. «Desarrollo global de software». En: *RAMA* (2014).

[39] Rafael Prikladnicki y Leonardo Pilatti. «Improving contextual skills in global software engineering: A corporate training experience». En: *IEEE International Conference on Global Software Engineering*. IEEE. 2008, págs. 239-243.

[40] Javier Saldaña-Ramos y col. «Skills and abilities for working in a global software development team: a competence model». En: *Journal of software: evolution and process* 26.3 (2014), págs. 329-338.

[41] Ken Schwaber y Jeff Sutherland. «La guía de scrum: La guía definitiva de scrum, las reglas del juego». En: *Recuperado de http://www. scrumguides. org/docs/scrumguide/v1/Scrum-Guide-ES. pdf* (2013).

[42] Kamalrufadillah Sutling y col. «Understanding of project manager competency in agile software development project: The taxonomy». En: *Information Science and Applications*. Springer, 2015, págs. 859-868.

[43] June M Verner y col. «Risks and risk mitigation in global software development: A tertiary study». En: *Information and Software Technology* 56.1 (2014), págs. 54-78.

[44] Aurora Vizcaíno, Félix García y Mario Piattini. «Visión general del desarrollo global de software». En: *International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC)* 1.1 (2015), págs. 8-22.

[45] Christiane Gresse Von Wangenheim, Rafael Savi y Adriano Ferreti Borgatto. «SCRUMIA—An educational game for teaching SCRUM in computing courses». En: *Journal of Systems and Software* 86.10 (2013), págs. 2675-2687.

[46] Wim Westera y col. «Artificial intelligence moving serious gaming: Presenting reusable game AI components». En: *Education and Information Technologies* 25.1 (2020), págs. 351-380.