

ANEXO I
PLAN DEL PROYECTO SOFTWARE

ÍNDICE DE CONTENIDO

Anexo I	
Plan del Proyecto Software.....	95
Lista de cambios.....	101
1. INTRODUCCIÓN.....	103
2. GESTIÓN DE RIESGOS.....	104
2.1. Clasificación de la gestión de riesgos.....	104
2.2. Elementos de la gestión de riesgos.....	105
2.3. Estimación de riesgos asociados al proyecto.....	108
2.3.1. Identificación de riesgos.....	108
2.3.2. Análisis de riesgos.....	109
2.3.3. Evaluación de riesgos.....	110
2.4. Control de riesgos asociados al proyecto.....	111
2.4.1. Planificación de la gestión de riesgos.....	111
2.4.2. Supervisión de riesgos.....	112
3. PLANIFICACIÓN TEMPORAL.....	113
3.1. Planificación inicial del proyecto.....	113
3.2. Planificación final del proyecto.....	119
3.3. Desviaciones en la planificación inicial.....	123
3.4. Descripción de las tareas.....	124
3.4.1. Estudio de la documentación previa.....	124
3.4.2. Enumeración de requisitos.....	124
3.4.3. Planificación temporal.....	124
3.4.4. Estudio preliminar de costes.....	125
3.4.5. Análisis de riesgos.....	125
3.4.6. Estudio de herramientas a utilizar.....	125
3.4.7. Requisitos tecnológicos.....	125
3.4.8. Traspaso del plugin a versión 3.5 de Eclipse.....	126
3.4.9. Incorporación de última versión de bibliotecas.....	126
3.4.10. Iteración 1.....	126
3.4.11. Iteración 2.....	127
3.4.12. Iteración 3.....	127
3.4.13. Iteración 4.....	128
3.4.14. Iteración 5.....	128
3.4.15. Iteración 6.....	129
3.4.16. Iteración 7.....	129
3.4.17. Documentación.....	130
4. ESTUDIO DE VIABILIDAD.....	130
4.1. Viabilidad técnica.....	131
4.2. Viabilidad legal.....	131

<u>4.3. Viabilidad económica.....</u>	<u>133</u>
<u>4.3.1. Análisis de costes.....</u>	<u>133</u>
<u>4.3.2. Análisis coste-beneficio.....</u>	<u>138</u>
<u>4.4. Viabilidad temporal.....</u>	<u>144</u>
<u>4.4.1. Estimación de Casos de Uso.....</u>	<u>144</u>

ÍNDICE DE ILUSTRACIONES

Ilustración 12: Diagrama de Gantt de la planificación inicial del proyecto.....	117
Ilustración 13: Diagrama de Gantt de la planificación final del proyecto.....	122

ÍNDICE DE TABLAS

Tabla 2: Categorías definidas para la clasificación de riesgos.....	109
Tabla 3: Escalas de medida definidas para la evaluación del análisis de riesgos.....	110
Tabla 4: Tabla de riesgos del proyecto.....	110
Tabla 5: Tabla de riesgos del proyecto priorizada.....	111
Tabla 6: Planificación inicial del proyecto.....	116
Tabla 7: Planificación final del proyecto.....	121
Tabla 8: Relación de licencias de las herramientas utilizadas.....	132
Tabla 9: Coste de las herramientas utilizadas.....	136
Tabla 10: Costes del proyecto.....	137
Tabla 11: Precios de las licencias comerciales de IntelliJIDEA.....	140
Tabla 12: Relación entre el precio y el número de licencias de la aplicación.....	141
Tabla 13: Valores utilizados para el cálculo del VAN.....	143
Tabla 14: Peso de los casos de uso.....	145
Tabla 15: Peso de los factores técnicos.....	146
Tabla 16: Peso de los factores de entorno.....	147

LISTA DE CAMBIOS

Número	Fecha	Descripción	Autor/es
0	11/04/11	Primera versión con introducción y comienzo de la planificación temporal.	Míryam Gómez San Martín Íñigo Mediavilla Saiz
1	23/04/11	Planificación temporal definitiva, desviaciones y primera versión del estudio de viabilidad.	Míryam Gómez San Martín Íñigo Mediavilla Saiz
2	08/05/11	Incorporada introducción y gestión de riesgos.	Míryam Gómez San Martín Íñigo Mediavilla Saiz
3	09/05/11	Descripción de tareas y viabilidad legal y técnica.	Míryam Gómez San Martín Íñigo Mediavilla Saiz
4	25/05/11	Incluida viabilidad temporal y económica.	Míryam Gómez San Martín Íñigo Mediavilla Saiz
5	06/06/11	Revisión final e incluidos pequeños detalles.	Míryam Gómez San Martín Íñigo Mediavilla Saiz

1. INTRODUCCIÓN

El Plan de Proyecto Software tiene como objetivo inicial establecer la identificación y el análisis de riesgos asociados al desarrollo software, los cuales pueden comprometer el futuro del proyecto y su desarrollo en unos plazos aceptables, es por esta razón que se han de tener siempre presentes. Finalmente, su propósito es el de proporcionar la información necesaria para llevar un control sobre el desarrollo del proyecto software.

Durante el proceso de desarrollo de un producto *software* una de las tareas más importantes que debe llevarse a cabo es la planificación, ya que esta se puede tomar como base para la toma de decisiones. El objetivo final de la planificación es determinar el tiempo necesario para la realización del proyecto con el fin de poder ajustarse lo máximo posible a este. Además ayuda a determinar los recursos necesarios y los costes que la elaboración del mismo lleva asociados consigo.

Realizaremos el estudio de la planificación teniendo en cuenta los aspectos que se van a indicar a continuación:

- Planificación temporal del proyecto: determina el calendario con el que vamos a poder representar de forma gráfica todas las tareas a realizar para la obtención del producto final. Se mostrará tanto la planificación inicial como el resultado final, con el objetivo de poder observar la evolución que esta ha sufrido. Asimismo se analizarán los aspectos más relevantes que hayan llevado a una alteración de dicha planificación.
- Estudio de la viabilidad del proyecto desde tres puntos de vista principalmente: el punto de vista temporal, el punto de vista económico y el punto de vista legal. El estudio de la viabilidad económica estará fundamentado en la técnica de análisis coste/beneficio. En el estudio correspondiente a la viabilidad legal se explicará y justificará la licencia elegida para la distribución del producto software.

2. GESTIÓN DE RIESGOS

Los objetivos de la gestión de riesgos son identificar, controlar y eliminar las fuentes de riesgo, antes de que empiecen a afectar al cumplimiento de los objetivos del proyecto. Es por ello que es muy importante el análisis de riesgos con el fin de identificar y poder evaluar la probabilidad de que estos ocurran para estimar el impacto, clasificarlos según la importancia de los mismos y establecer con ello un plan de contingencia en caso de que el problema se presente.

Los riesgos involucran siempre dos características, estas son las siguientes:

Incertidumbre: el acontecimiento que caracteriza al riesgo puede o no ocurrir.

Pérdida potencial: si el riesgo termina por convertirse en una realidad, ocurrirán las consecuencias no deseadas o pérdidas.

Para poder cuantificar el nivel de incertidumbre y el grado de pérdidas asociado con cada riesgo se consideran diferentes categorías de riesgos, a continuación serán detalladas.

2.1. Clasificación de la gestión de riesgos

En función del nivel de incertidumbre y el grado de pérdidas asociado los riesgos se pueden clasificar en las siguientes categorías:

Riesgos del Proyecto

- Amenazan al Plan del Proyecto, es decir, afectan a la planificación temporal, al coste y a la calidad del proyecto.
- Identifican problemas potenciales de presupuesto, de calendario, de personal, de recursos, de cliente, etc.

Riesgos Técnicos

- Amenazan la calidad y la planificación temporal del producto software que se producirá.
- Identifican posibles problemas de incertidumbre técnica, ambigüedad en la especificación, en diseño, implementación, obsolescencia técnica o tecnología

puntera, interfaz, verificación y mantenimiento, etc.

Riesgos del Negocio

- Amenaza la viabilidad del software que se construirá.
- Los principales riesgos de negocio son:
 - riesgo de mercado: cuando el producto es demasiado bueno.
 - riesgo estratégico: en el caso de productos que no encajan.
 - riesgo de ventas: si el producto es poco vendible.
 - riesgo de presupuesto: cuando el producto esté fuera de presupuesto.

Se puede hacer otra categorización de los riesgos diferente en función de la facilidad que tienen para ser detectados. En este caso existen los siguientes:

Riesgos conocidos: son aquellos que se pueden predecir después de realizar una evaluación del plan del proyecto, del entorno técnico y de otras fuentes fiables de información.

Riesgos predecibles: cuando es posible realizar la extrapolación de la experiencia de proyectos anteriores.

Riesgos impredecibles: son aquellos que pueden ocurrir, pero es extremadamente difícil identificarlos por adelantado.

2.2. Elementos de la gestión de riesgos

La gestión continuada de los riesgos permite aumentar su eficiencia, para que esto sea posible tendremos que evaluar continuamente lo que pueda ir mal, determinar qué riesgos son importantes para implementar estrategias para resolverlos en caso de que se produjesen y asegurarnos de la eficacia de dichas estrategias.

Los elementos de la gestión de riesgos son los siguientes:

Estimación de riesgos

- *Identificación de riesgos:* lista de riesgos potenciales que pueden afectar a la planificación del proyecto.

Las incertidumbres sobre diferentes características del proyecto que puedan

existir se transforman en riesgos que pueden ser descritos y medidos. Un método para identificar los riesgos es crear una lista de comprobación de elementos de riesgo que podría contener dos categorías de riesgos:

Riesgos específicos del producto: se los puede identificar con un buen conocimiento de la tecnología, el personal y el entorno específico del software, para ello se debe examinar el plan del proyecto y la declaración del ámbito del software.

Riesgos genéricos: son comunes a todos los proyectos de software.

Para identificarlos se crean las siguientes subcategorías:

- Tamaño del producto.
 - Impacto en el negocio.
 - Características del cliente.
 - Definición del proceso.
 - Entorno de desarrollo.
 - Tecnología a construir.
 - Tamaño y experiencia de la plantilla.
- *Análisis de riesgos:* medición de la probabilidad y el impacto de cada riesgo, y los niveles de riesgo de los métodos alternativos.

Es el proceso de examinar los riesgos en detalle para determinar su extensión, sus interrelaciones y su importancia.

Las actividades básicas son:

Evaluación: mejor comprensión del riesgo.

Se cuantifican, en lo posible, los siguientes conceptos:

Impacto: pérdida que ocasiona el riesgo. Consecuencias de los problemas asociados con el riesgo. Los factores que afectan al impacto son:

- naturaleza: problemas potenciales que se pueden producir en caso de ocurrir.

- alcance: combina la severidad con su distribución global.
- duración: combina el momento en el que se sentirá su impacto y la duración del mismo.

Probabilidad: probabilidad de que ocurra el riesgo.

Marco de tiempo: periodo de tiempo en el que es posible mitigar el riesgo.

Clasificación: se clasifican los riesgos para entender su naturaleza y elaborar planes de mitigación.

- *Evaluación de riesgos:* lista de riesgos ordenados por su impacto y por su probabilidad de ocurrencia para determinar cuáles se deben solucionar antes y a cuáles hay que asignarles más recursos.

Los riesgos pueden ordenarse según la magnitud de la exposición al riesgo:

$$[r_i, l_i, x_i]$$

donde;

r_i : riesgo.

l_i : probabilidad del riesgo.

x_i : magnitud del impacto del riesgo.

Las condiciones y prioridades pueden cambiar a lo largo del proyecto por lo que es importante destacar que el análisis y la asignación de prioridades debe realizarse de manera continua aprovechando la información disponible en cada momento.

Control de riesgos

- *Planificación de la gestión de riesgos:* plan para tratar cada uno de los riesgos significativos.
- *Supervisión de riesgos:* comprobación del progreso del control de un riesgo e identificación de la aparición de nuevos riesgos.

2.3. Estimación de riesgos asociados al proyecto

Este apartado esta dedicado a la estimación de los riesgos asociados a nuestro proyecto, en él se identificarán dichos riesgos para posteriormente pasar a ser analizados y poder así evaluarlos y clasificarlos con el objetivo final de poder establecer un orden de prioridades sobre los mismos.

2.3.1. Identificación de riesgos

A continuación se enumera una serie de riesgos que han surgido al evaluar futuros problemas y contingencias del proyecto a tratar. Dichas dificultades, surgen de las distintas reuniones con el tutor del proyecto y de una propia autocrítica de las personas involucradas, considerando sobre todo qué aspectos eran más negativos y en qué proporción afectaban al desarrollo satisfactorio del proyecto.

- Riesgo 1: El equipo de desarrollo tiene escasa experiencia en el proyecto a desarrollar (desarrollo de plugin, utilización de nuevas herramientas, bibliotecas de Eclipse y bibliotecas externas necesarias) esto hace que se requiera una inversión significativa en la curva de aprendizaje y en la propia formación de los desarrolladores.
- Riesgo 2: La planificación temporal estimada al principio del proyecto puede ser irreal o la fecha de entrega estar muy ajustada, pudiendo acarrear retrasos considerables en el desarrollo del proyecto.
- Riesgo 3: Se podría producir cambios en los requisitos iniciales por ser estos inadecuados (características que funcionan de forma distinta a como se esperaba, características necesarias sobre las que nadie pensó) teniendo un efecto muy negativo en etapas avanzadas de desarrollo.
- Riesgo 4: Se produce un solapamiento temporal con otras actividades que los desarrolladores implicados realizan, como son: trabajo, cursos, actividades personales, etc.
- Riesgo 5: Limitaciones en cuanto a disponibilidad de documentación referente al desarrollo de plugins para Eclipse (herramientas, bibliotecas, etc).
- Riesgo 6: La versión del plugin sobre la que se parte para realizar el desarrollo de este proyecto puede presentar fallos y características que no funcionan

como se esperan y que van a ser localizadas durante el desarrollo del mismo y tendrán que ser corregidas.

- Riesgo 7: Necesidad de realizar modificaciones internas durante el desarrollo del proyecto en la aplicación que afecten al propio diseño de la misma y que destaque por su complejidad.

2.3.2. Análisis de riesgos

Una sencilla técnica para llevar a cabo el análisis de riesgos será la utilización de una *tabla de riesgos*. En la primera columna de esta tabla se listarán todos los riesgos del proyecto que han sido identificados en el apartado anterior. La segunda columna servirá para clasificar cada uno de estos riesgos según la categoría a la que pertenecen. Por último, las siguientes dos columnas se utilizarán para valorar la probabilidad y el impacto de los diferentes riesgos registrados en la primera columna de la tabla. Opcionalmente se podría añadir una última columna para determinar el marco de tiempo asociado al riesgo, es decir, el periodo de tiempo en el que es posible mitigar el riesgo.

Una vez definida la estructura de la *tabla de riesgos* daremos paso a indicar las categorías a utilizar para la clasificación de los riesgos y a definir la escala de medidas a utilizar.

Para la clasificación de los riesgos vamos a utilizar las categorías que a continuación se muestran y que hemos definido de la siguiente forma:

Categoría	Descripción
PS	Tamaño del producto
BU	Impacto en el negocio
TE	Tecnología
ST	Personal
DE	Entorno de desarrollo

Tabla 2: Categorías definidas para la clasificación de riesgos

Para la especificación de estas medidas hemos elegido los términos cualitativos que a continuación se indican, junto con su correspondiente interpretación cuantitativa, todo ello se recoge en la tabla siguiente.

Atributo	Valor	Descripción
Probabilidad	Muy Probable	>70%
	Probable	entre 30% y 70%
	Improbable	<30%
Impacto	Catastrófico	Pérdida del sistema. Coste >50%
	Crítico	Recuperación de la capacidad operativa. Coste >10% (Coste <50%)
	Marginal	Coste <10%
	Despreciable	Coste ≈ 0
Marco de Tiempo	Corto Plazo	15 días
	Medio Plazo	de 1 a 2 meses.
	Largo Plazo	más de 2 meses.

Tabla 3: Escalas de medida definidas para la evaluación del análisis de riesgos

A partir de los riesgos identificados en el apartado anterior y en base a los criterios que se acaban de definir, a continuación se presenta la *tabla de riesgos* relativa a nuestro proyecto:

Riesgos	Categoría	Probabilidad	Impacto
Riesgo 1	DE y ST	Muy Probable (80%)	Crítico
Riesgo 2	BU	Improbable (20%)	Marginal
Riesgo 3	PS	Muy Probable (70%)	Marginal
Riesgo 4	ST	Probable (30%)	Despreciable
Riesgo 5	TE	Muy Probable (80%)	Crítico
Riesgo 6	DE y PS	Probable (40%)	Marginal
Riesgo 7	DE y PS	Probable (50%)	Marginal

Tabla 4: Tabla de riesgos del proyecto

2.3.3. Evaluación de riesgos

Una vez que ya se dispone de la *tabla de riesgos* deberemos de ordenarla por probabilidad y por impacto, de tal forma que los riesgos de elevada probabilidad y alto impacto pasen a las primera filas de la tabla, mientras que los riesgos de baja probabilidad caerán a la parte inferior de la misma. Esto nos permitirá establecer un orden de prioridades sobre los riesgos asociados al proyecto, lo que facilitará la definición de una linea de corte

para determinar qué riesgos deberán seguir gestionándose y cuáles no.

A continuación se muestra la *tabla de riesgos* relativa a nuestro proyecto ordenada en función de las prioridades sobre los riesgos asociados al proyecto:

Riesgos	Categoría	Probabilidad	Impacto
Riesgo 1	DE y ST	Muy Probable (80%)	Crítico
Riesgo 5	TE	Muy Probable (80%)	Crítico
Riesgo 3	PS	Muy Probable (70%)	Marginal
Riesgo 7	DE y PS	Probable (50%)	Marginal
Riesgo 6	DE y PS	Probable (40%)	Marginal
Riesgo 4	ST	Probable (30%)	Despreciable
Riesgo 2	BU	Improbable (20%)	Marginal

Tabla 5: Tabla de riesgos del proyecto priorizada

2.4. Control de riesgos asociados al proyecto

Las actividades que conforman la gestión de riesgos del proyecto descritas hasta ahora tienen un único objetivo final y este no es otro que el de ayudarnos a desarrollar una estrategia para tratar eficazmente los riesgos identificados. Esto conlleva la consideración de tres aspectos fundamentales: evitar el riesgo siempre que resulte posible, supervisar el riesgo, y por último gestionar el riesgo y establecer unos buenos planes de contingencia. A continuación detallaremos cada uno de estos aspectos.

2.4.1. Planificación de la gestión de riesgos

En nuestro caso hemos adoptado un enfoque proactivo frente al riesgo, evitando el riesgo como la mejor estrategia a seguir. Esto se consigue definiendo un plan de reducción del riesgo. Este plan contará con los siguientes aspectos:

- Utilizar una herramienta para el control de tareas con el fin de identificar cada tarea, la persona a la que está asociada en cada momento, así como reflejar en ella el tiempo estimado para su finalización y también el tiempo empleado, permitiendo la trazabilidad y medición de la misma. Con ella además de realizar la gestión de tareas también se realizará lo propio con la gestión de errores.
- Aumentar la batería de test disponibles para la aplicación con el fin de cubrir

el mayor número de casos que se puedan dar, con el objetivo de garantizar la consistencia y compatibilidad funcional del programa. También sería adecuado realizar test gráficos y pruebas de estrés.

- Definir un método de estimación de tareas, por ejemplo para cada tarea se estimará el mejor caso, el peor y el más probable.
- También se deberán recoger datos históricos para ver la evolución del trabajo (por ejemplo, mediante el estado de las tareas: abierta, en progreso, en stand by, terminada) así como también de los errores registrados (si se han detectado por los programadores y en qué momento –revisión, compilación, pruebas-, o si lo ha detectado el usuario). Gracias a ellos podemos estimar mucho mejor ya que nos podemos basar en experiencias pasadas, de igual modo nos permitirá justificar planificaciones.
- Definir plantillas de documentación y establecer mecanismos para asegurar que los documentos se vayan generando puntualmente.
- Realizar una buena documentación técnica para que aquel que desee realizar cambios futuros en la aplicación encuentre en ella una buena fuente de información, siendo de gran ayuda debida a la escasa documentación que existe sobre el desarrollo de plugins de Eclipse.
- Llevar a cabo revisiones en equipo del proyecto con el fin de que todas las personas, alumnos y tutor, vayan siguiendo el proceso y la evolución del mismo. Con ello conseguiremos poner al día la planificación y determinar la velocidad del proyecto. Además también será importante que cada uno de ellos deje sus impresiones del proyecto con el propósito de realizar una mejora continua del mismo.

2.4.2. Supervisión de riesgos

La supervisión de riesgos [Pressman 2006] asociados a nuestro proyecto supone:

- Detectar la ocurrencia de un riesgo que haya sido previsto.
- Asegurar que los pasos de reducción definidos para cada riesgo se estén aplicando correctamente.

- Identificar la aparición de nuevos riesgos que no hayan sido previstos.
- Recopilar información con el fin de que esta pueda ser de utilidad de cara al análisis de nuevos riesgos que se detecten o de riesgos de futuros proyectos.

3. PLANIFICACIÓN TEMPORAL

En este apartado se recoge la planificación temporal del proyecto, es decir, la elaboración del calendario o programa de tiempos. El calendario del proyecto es una representación gráfica de todas las actividades del proyecto, necesarias para producir el resultado final, permitiendo al desarrollador del proyecto coordinar de forma efectiva el transcurso del mismo.

De forma general se puede ver la planificación temporal como la división del proyecto en actividades, cada una de las cuales va acompañada de una estimación del tiempo destinado a su realización. El objetivo de la planificación temporal es coordinar estas tareas.

En este apartado se muestra la planificación temporal que hemos empleado para el desarrollo del presente proyecto. Para ello hemos utilizado la técnica gráfica de *Diagramas de Gantt*, que se trata de una representación gráfica de las tareas sobre una escala de tiempos. Las tareas se representan en forma de barra sobre dicha escala manteniendo la relación de proporcionalidad entre sus duraciones, su representación gráfica y su posición respecto del punto origen del proyecto.

A continuación veremos tanto la planificación inicial, estimada al comienzo del proyecto, como la evolución temporal del proyecto, es decir, el resultado final.

3.1. Planificación inicial del proyecto

La primera planificación temporal se realizó después de las primeras reuniones con el tutor en las que se especificaron los objetivos del proyecto en forma de requisitos, tanto funcionales como no funcionales.

La intención al elaborar esta primera planificación temporal fue la de no dejar olvidada ninguna de las tareas principales. En cuanto a la granularidad se trató de que las tareas se especificaran de una manera suficientemente exhaustiva como para no cometer desviaciones muy graves en su acotación temporal.

El proceso de desarrolló se decidió que se conformara en base a ciclos conocidos como iteraciones. Al final de cada ciclo se realizaría una reunión con el tutor en el que se revisarían las tareas implementadas y se corregirían aspectos que el tutor junto con los alumnos consideraran mejorables o erróneos. Este modelo de iteraciones similar al postulado por algunas metodologías ágiles [Schwaber 2004] se consideró el más adecuado para el desarrollo. Es un modelo que facilita estar centrado exclusivamente en un conjunto de tareas lo que mejora el desarrollo de las mismas. Las tareas a desarrollar deben ser las de más valor para el proyecto así se facilita la priorización y visibilidad del mismo de cara al usuario. Finalmente, la revisión periódica de la evolución del proyecto evita que el desarrollo se desvíe de los intereses principales del usuario y permite al usuario aportar nuevas sugerencias de forma temprana.

En la primera planificación se optó por posponer la elaboración del grueso de la documentación para el final del proyecto. Esta decisión se tomó debido a que se consideró que la mayoría de los aspectos a documentar no estarían definidos hasta el final del proyecto y escribirlos desde el principio supondría trabajo repetido.

Tarea	Duración	Fecha Inicio	Fecha Fin
Estudio de documentación previa	3 días	07/01/2011	12/01/2011
Enumeración de requisitos funcionales	2 días	12/01/2011	14/01/2011
Planificación temporal	1 días	14/01/2011	15/01/2011
Estudio preliminar de costes y de viabilidad	1 días	17/01/2011	18/01/2011
Estudio de herramientas a utilizar	1 días	18/01/2011	19/01/2011
Traspaso del plugin a versión 3.5 de Eclipse	2 días	19/01/2011	21/01/2011
Incorporación de última versión de bibliotecas	2 días	21/01/2011	25/01/2011
Iteración 1	9 días	26/01/2011	08/02/2011
Separación del proyecto de tests	3 días	26/01/2011	29/01/2011
Automatización de la construcción del proyecto	5 días	31/01/2011	05/02/2011
Configuración de proceso de build diario	1 días	07/02/2011	08/02/2011
Prototipado de vista de catálogo de refactorizaciones	1 días	07/02/2011	08/02/2011
Iteración 2	9 días	09/02/2011	22/02/2011
Implementación del modelo del dominio de la clasificación de refactorizaciones	4 días	09/02/2011	15/02/2011
Filtrado de Clasificaciones	5 días	15/02/2011	22/02/2011
Actualización del XML de las refactorizaciones	3 días	15/02/2011	18/02/2011
Iteración 3	9 días	23/02/2011	08/03/2011
Versión de línea de comandos del catálogo de clasificaciones	2 días	23/02/2011	25/02/2011
Primera versión de la vista del catálogo de refactorizaciones en Eclipse	4 días	25/02/2011	03/03/2011
XML de las clasificaciones	3 días	03/03/2011	08/03/2011
Iteración 4	9 días	09/03/2011	22/03/2011
Instalación de plugin desde Internet	2 días	09/03/2011	11/03/2011
Editor de clasificaciones desde Eclipse	3 días	11/03/2011	16/03/2011
Pestañas con información adicional en vista del catálogo	3 días	16/03/2011	19/03/2011
Prototipado de la ayuda a la creación de refactorizaciones	1 días	21/03/2011	22/03/2011

Iteración 5	9 días	22/03/2011	02/04/2011
Debate y prototipado sobre versiones de las refactorizaciones	2 días	22/03/2011	24/03/2011
Implementación de la obtención de la metainformación de entradas, predicados y acciones	4 días	24/03/2011	30/03/2011
Implementación de la búsqueda avanzada en la ayuda	3 días	30/03/2011	02/04/2011
Iteración 6	9 días	05/04/2011	16/04/2011
Implementación de versionado de refactorizaciones	4 días	05/04/2011	09/04/2011
Implementación de versionado de clasificaciones	3 días	11/04/2011	14/04/2011
Implementación de previsualización de resultados de aplicar refactorización	2 días	14/04/2011	16/04/2011
Documentación	28 días	18/04/2011	26/05/2011
Manual de usuario	5 días	18/04/2011	23/04/2011
Documentación técnica	6 días	25/04/2011	03/05/2011
Memoria principal del proyecto	12 días	03/05/2011	19/05/2011
Anexos	5 días	19/05/2011	26/05/2011

Tabla 6: Planificación inicial del proyecto

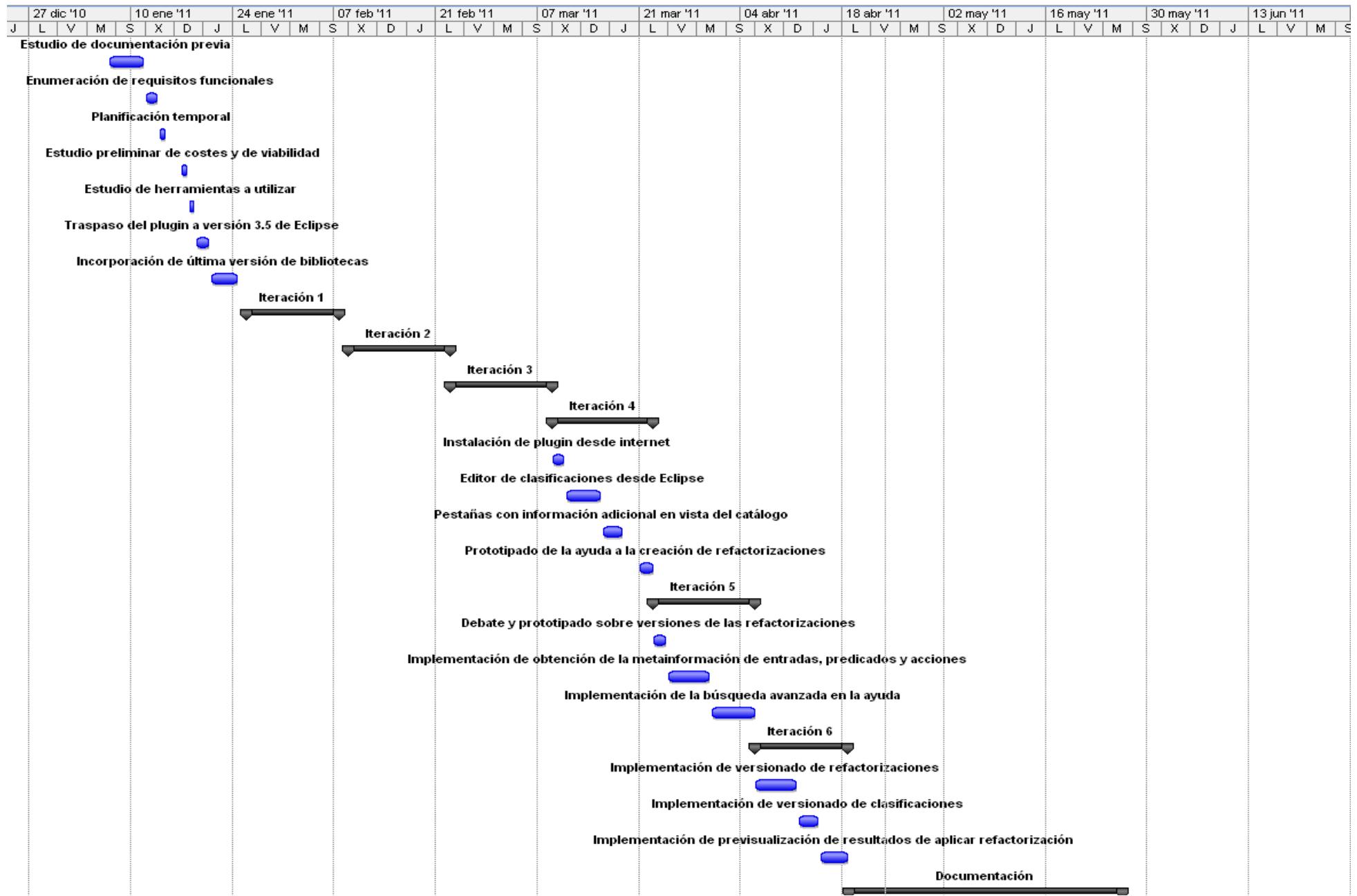


Ilustración 12: Diagrama de Gantt de la planificación inicial del proyecto

3.2. Planificación final del proyecto

En la planificación definitiva ya aparecen reflejados todas las tareas desarrolladas durante el proyecto y la fecha en la que se realizaron.

Tarea	Duración	Fecha Inicio	Fecha Fin
Estudio de documentación previa	3 días	07/01/2011	12/01/2011
Enumeración de requisitos funcionales	2 días	12/01/2011	14/01/2011
Planificación temporal	1 días	14/01/2011	15/01/2011
Estudio preliminar de costes y de viabilidad	1 días	17/01/2011	18/01/2011
Análisis de riesgos	1 días	18/01/2011	19/01/2011
Estudio de herramientas a utilizar	1 días	18/01/2011	19/01/2011
Requisitos tecnológicos	1 días	19/01/2010	20/01/2010
Traspaso del plugin a versión 3.5 de Eclipse	2 días	19/01/2011	21/01/2011
Incorporación de última versión de bibliotecas	2 días	21/01/2011	25/01/2011
Iteración 1	9 días	26/01/2011	08/02/2011
Separación del proyecto de tests	2 días	26/01/2011	28/01/2011
Automatización de la construcción del proyecto (1)	2 días	28/01/2011	01/02/2011
Prototipado de vista de catálogo de refactorizaciones	1 días	01/02/2011	02/02/2011
Implementación del modelo del dominio de la clasificación de refactorizaciones	4 días	28/01/2011	03/02/2011
Filtrado de Clasificaciones	2 días	03/02/2011	05/02/2011
Cambio Interno - Sustituir constantes enteras			
RefactoringConstants.Action/Precondition n/Postcondition por Enum	1 días	07/02/2011	08/02/2011
Iteración 2	9 días	09/02/2011	22/02/2011
Automatización de la construcción del proyecto (2)	1 días	09/02/2011	10/02/2011
Configuración de proceso de build diario	1 días	10/02/2011	11/02/2011
XML de las clasificaciones	3 días	11/02/2011	16/02/2011
Actualización del XML de las refactorizaciones	2 días	16/02/2011	18/02/2011
Crear vista de catalogo - Asignar categorias al crear/editar una refact.	2 días	18/02/2011	22/02/2011

Tarea	Duración	Fecha Inicio	Fecha Fin
Iteración 3	9 días	22/02/2011	05/03/2011
Versión de línea de comandos del catálogo de clasificaciones	2 días	22/02/2011	24/02/2011
Primera versión de la vista del catálogo de refactorizaciones en Eclipse	2 días	24/02/2011	26/02/2011
Permitir agregar palabras claves a los metadatos de una refactorización	2 días	28/02/2011	02/03/2011
Añadir al xml de las refactorizaciones el scope al que pertenecen	1 días	02/03/2011	03/03/2011
Vista Catálogo - Agregar botón de refrescar refactorizaciones	2 días	03/03/2011	05/03/2011
Configurar servidor virtual en amazon para servidor de build y métricas	4 días	22/02/2011	26/02/2011
Instalación de plugin desde internet	5 días	25/02/2011	04/03/2011
Iteración 4	9 días	07/03/2011	18/03/2011
Editor de clasificaciones desde Eclipse	3 días	07/03/2011	10/03/2011
Prototipado de la ayuda a la creación de refactorizaciones	1 días	10/03/2011	11/03/2011
Pestañas con información adicional en vista del catálogo	1 días	11/03/2011	12/03/2011
Vista Catalogo - Filtrado desde interfaz	1 días	14/03/2011	15/03/2011
Vista Catálogo - Añadir un Examples Tab al FolderTab	1 días	15/03/2011	16/03/2011
Bug - DynamicRefactoring Wizard - No muestra el javadoc de los elementos	1 días	16/03/2011	17/03/2011
Bug - Error al intentar aplicar la refactorización RenameParameter de un método	1 días	17/03/2011	18/03/2011
Bug - Error al intentar aplicar RenameField	1 días	17/03/2011	18/03/2011
Iteración 5	9 días	21/03/2011	01/04/2011
Debate y prototipado sobre versiones de las refactorizaciones	2 días	21/03/2011	23/03/2011
Implementación de obtención de la metainformación de entradas, predicados y acciones	4 días	23/03/2011	29/03/2011
Implementación de la búsqueda avanzada en la ayuda	2 días	29/03/2011	31/03/2011
Incorporar certificado para la instalación del plugin	1 días	31/03/2011	01/04/2011

Tarea	Duración	Fecha Inicio	Fecha Fin
Bug Wizard - Input Configuration (Step 2 of 7) - flechas orden lista	2 días	25/03/2011	29/03/2011
Cambio Interno - Sustituir InputParameters como HashMap por objeto con atributos	2 días	29/03/2011	31/03/2011
Iteración 6	9 días	04/04/2011	15/04/2011
Implementación de versionado de refactorizaciones	3 días	04/04/2011	07/04/2011
Cambio interno : Sustituir Ambiguous Parameter como Mapa por objeto perteneciente a una precondition, accion o postcondicion	1 días	07/04/2011	08/04/2011
Cambio interno: Repositorio único de refactorizaciones accesibles desde un directorio independiente del workspace	4 días	08/04/2011	14/04/2011
Cambio interno : Hacer las refactorizaciones accesibles desde un directorio independiente del workspace	1 días	14/04/2011	15/04/2011
Iteración 7	8 días	18/04/2011	28/04/2011
Habilitar un servidor web o ftp con ip fija para instalacion del plugin desde internet	2 días	18/04/2011	20/04/2011
Bug Wizard: (Step 1 of 7) Interfaz se descuadra ante modificaciones de la ventana	2 días	20/04/2011	22/04/2011
Versionado de clasificaciones	2 días	22/04/2011	26/04/2011
Mejoras al editor de clasificaciones	2 días	26/04/2011	28/04/2011
Documentación	28 días	29/04/2011	08/06/2011
Manual de usuario	5 días	29/04/2011	06/05/2011
Documentación técnica	6 días	06/05/2011	14/05/2011
Memoria principal del proyecto	12 días	16/05/2011	01/06/2011
Anexos	5 días	01/06/2011	08/06/2011

Tabla 7: Planificación final del proyecto

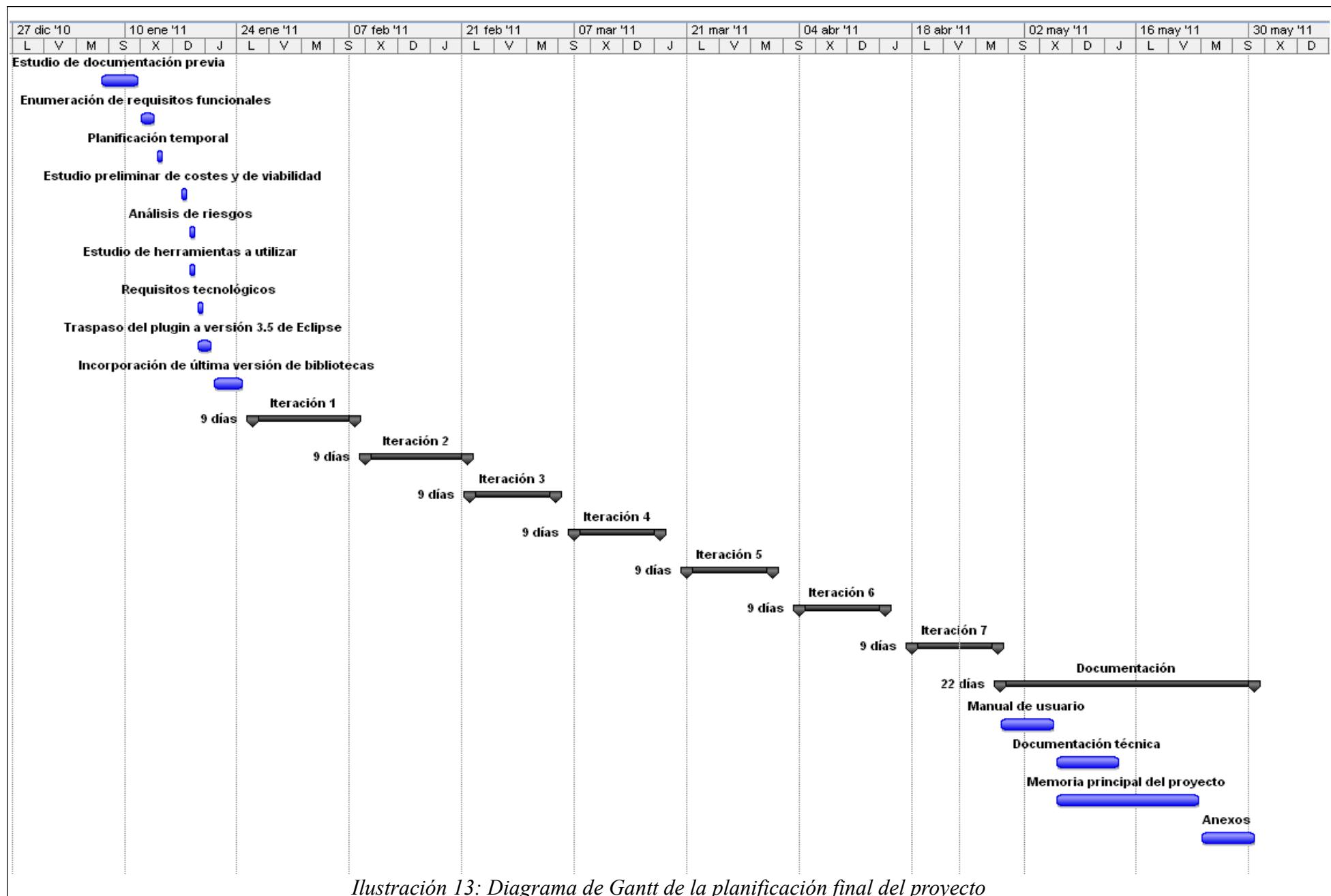


Ilustración 13: Diagrama de Gantt de la planificación final del proyecto

3.3. Desviaciones en la planificación inicial

Como se puede observar si se realiza una comparativa entre las dos planificaciones temporales inicial y final del proyecto se han producido numerosas modificaciones al plan original. El objetivo del plan inicial era servir de guía de la evolución del proyecto con el fin de priorizar aquellas tareas de mayor importancia, dando un cierto sentido al orden en que se deberían de hacer efectivas estas. En ningún momento se tomó el plan de proyecto inicial como una lista de obligaciones inflexibles dado que se era consciente de que el desarrollo del proyecto impondría modificaciones a esa previsión inicial. A continuación se describirán las principales desviaciones que se han producido por ese discurrir del proyecto y las causas que las provocaron.

Las variaciones han consistido no sólo en desviaciones temporales, sino que también ha habido cambios en las características a implementar, tareas que se habían planificado y se han descartado por no considerarse tan importantes o por las dificultades que presentaba su implementación. Por ejemplo, la posibilidad de previsualizar los cambios producidos en un código fuente a consecuencia de la ejecución de una refactorización.

Por otra parte otras tareas que no habían sido consideradas en la planificación inicial han sido agregadas. Ejemplo de esto son ciertas mejoras internas que se han efectuado y que fue imposible planificar inicialmente por desconocimiento del código fuente. Otro tipo de tarea que era muy difícil de planificar inicialmente era la resolución de errores encontrados en el proyecto. Al ser esta la tercera versión del plugin se han encontrado algunos defectos heredados de versiones previas del plugin que han sido necesarios corregir. Algunos de los defectos más importantes corregidos aparecen reflejados en la planificación definitiva.

Otro apartado en el que la variación ha sido mayor es el proceso de construcción del proyecto. Este proceso si aparece reflejado en la primera versión de la planificación pero en ese momento se consideró que sería sencillo de implementar por lo que se planteó que podría quedar terminado para la primera iteración. Sin embargo, la falta de experiencia de los alumnos con la herramienta Maven ha hecho que esa configuración inicial se prolongara a lo largo de varias iteraciones. Además la flexibilidad proporcionada por la herramienta ha hecho que posteriormente se hayan ido agregando nuevas mejoras y funcionalidades adicionales que se consideraba que aportaban valor añadido al proyecto y que se han visto facilitadas por la herramienta. Ejemplos de estas nuevas funcionalidades ha sido la posibilidad de firmar el plugin de forma automática, de generar la cobertura de las pruebas tras cada construcción, la generación del repositorio de instalación del plugin o la

comprobación de métricas sobre el código gracias a Sonar.

En las fases finales del desarrollo se produjo un pequeño giro en la planificación. Debido a la inestabilidad de algunos apartados del plugin se decidió congelar el desarrollo de nuevas características de forma anticipada, previamente a lo que estaba previsto en la planificación inicial. Esto se puede comprobar en el hecho de que las últimas iteraciones están copadas principalmente por modificaciones de diseño y corrección de errores en la planificación definitiva.

Finalmente hay que destacar que todo el desarrollo del plan del proyecto se ha visto facilitado por la herramienta Fogbugz, la cual ya fue destacada en el apartado de *Técnicas y Herramientas de la Memoria*. Esta herramienta ha permitido llevar un control detallado de las tareas y ha facilitado la comunicación entre los miembros del equipo. Además ha sido de gran ayuda a la hora de realizar esta retrospectiva. Gracias a ella y a la labor de documentación de las tareas efectuada por los desarrolladores, se ha encontrado toda la evolución del proyecto reflejada en su página web cuando se ha necesitado para la elaboración de esta sección.

3.4. Descripción de las tareas

3.4.1. Estudio de la documentación previa

Los primeros días de trabajo se realiza un proceso de familiarización con el proyecto. En este periodo se estudia la documentación proporcionada por versiones anteriores del proyecto y se busca información de otras aplicaciones con funcionalidades semejantes. El objetivo es tener una imagen global del trabajo que se tiene que realizar.

3.4.2. Enumeración de requisitos

Se definen los requisitos del proyecto tanto funcionales como no funcionales. Consiste en definitiva en marcar los objetivos del proyecto.

3.4.3. Planificación temporal

Consiste en contextualizar las tareas a desarrollar del proyecto en el tiempo, decidiendo cuánto tiempo se dedicará a cada una de ellas y en qué momento se llevarán a cabo. De este modo se priorizan las importantes y se da sentido al orden de desarrollo de

características en base a cuales son dependientes de otras.

3.4.4. Estudio preliminar de costes

Consiste en plantearse un primer estudio de los costes y de la viabilidad del proyecto basándose en las funcionalidades que se van a obtener, en la competencia y en las necesidades del mercado. Se debe decidir y argumentar el seguir adelante con el proyecto. El estudio de viabilidad tendrá que ser revisado en la documentación al final del proyecto. Esto se debe a que a lo largo del proyecto se han podido utilizar algunas herramientas que no se pensaba adquirir en un principio.

3.4.5. Análisis de riesgos

En esta fase se tienen en cuenta los distintos riesgos conocidos y vigentes en el proyecto. El conjunto de los riesgos se ordena por importancia y se proponer de cada uno de ellos acciones específicas para su mitigación.

3.4.6. Estudio de herramientas a utilizar

Se estudian las necesidades de herramientas software para el desarrollo del proyecto de forma genérica. En base a ese estudio se toman las necesidades y por cada una se sopesan las distintas alternativas comerciales para escoger la que más se adapte a las necesidades del proyecto. En el caso de este proyecto las herramientas más necesarias a estudiar al comienzo del proyecto fueron el sistema de control de versiones a escoger, el de gestión de tareas y bugs y el sistema de construcción del proyecto a partir del código fuente.

3.4.7. Requisitos tecnológicos

En este apartado se plantean todas las técnicas, las herramientas y los entornos de desarrollo que se van a utilizar en la realización del proyecto. Se va a emplear más tiempo que para el resto de los requisitos debido a la necesidad de buscar las ventajas de utilizar determinadas herramientas frente a otras, para lo cual se necesita un estudio de las mismas. Se incluye también en este periodo el tiempo utilizado en la instalación de estos programas.

3.4.8. Traspaso del plugin a versión 3.5 de Eclipse

La última versión del plugin fue desarrollada sobre la versión 3.3 de Eclipse. Con la intención de hacer la nueva versión del plugin compatible con las últimas de Eclipse (3.5 en adelante), fue necesaria realizar una serie de ajustes de compatibilidad en el código fuente.

3.4.9. Incorporación de última versión de bibliotecas

De forma similar a lo que ocurría en la tarea anterior, la evolución del plugin se había congelado desde el proyecto anterior mientras que la de las bibliotecas MOON y JavaMOON había incorporado cambios importantes. De nuevo fue necesario realizar ciertos cambios para hacer posible la integración de las últimas versiones de las bibliotecas. Este es un proceso a destacar dado que las últimas versiones de dichas bibliotecas han incorporado modificaciones de gran importancia que han posibilitado pasar de permitir refactorizaciones exclusivamente sobre proyectos pequeños a permitir refactorizaciones sobre proyectos reales con miles de líneas de código.

También es necesario reseñar que este proceso se realizó en más ocasiones a lo largo del proyecto, aunque sólo se ha marcado como tarea al principio del proyecto por el mayor tiempo que llevó debido a que los cambios en las bibliotecas eran mayores y a que al ser la primera vez que se realizaba el proceso de integración implicó una mayor dificultad.

Además con las nuevas posibilidades abiertas que ofrecía las bibliotecas fue necesario incorporar otros cambios en determinadas partes del plugin, para adaptarlo para permitir hacer uso de esas nuevas opciones que las bibliotecas abrían.

3.4.10. Iteración 1

En esta primera iteración como parte de la configuración del proceso de construcción automático del proyecto se separaron los tests creando un proyecto de Eclipse específico para ellos. Además se consiguió configurar Apache Maven con tycho[Tycho n.d.] para que se descargara sus dependencias de un repositorio en Internet y se compilara correctamente.

En esta fase ya se empezó a desarrollar la vista del catálogo de refactorizaciones que se continuaría desarrollando en iteraciones posteriores. Los pasos en ese sentido dados en esta fase fueron: el prototipado de la interfaz de dicha vista, la implementación de su modelo del dominio y también el modelo de filtrado por varios criterios.

Como último aspecto destacable en esta fase también se realizó una refactorización con el objetivo de eliminar código dependiente de las constantes enteras que representaban a las precondiciones, acciones y postcondiciones por una enumeración. Esto permitió en fases posteriores de la aplicación simplificar el diseño del código.

3.4.11. Iteración 2

En esta segunda iteración se continuó con apartados del proyecto ya tocados en la primera fase. En el apartado de la configuración de la construcción del plugin se consiguió que los tests se ejecutaran como parte del proceso y que se generaran las métricas del proyecto con Sonar [Sonar n.d.]. Además se configuró Hudson [Hudson Continuous Integration n.d.] para que ejecutara de forma diaria el proceso de construcción completo a partir del contenido de la rama principal, master, en el repositorio del proyecto en Github [GitHub n.d.].

Dentro del apartado de la vista del catálogo de refactorizaciones se creó el soporte XML para las clasificaciones y además se modificó el de las refactorizaciones para adaptarse al catálogo. También se modificó la interfaz gráfica de creación de refactorizaciones para permitir agregar clasificaciones a las refactorizaciones creadas.

3.4.12. Iteración 3

Se continua con la configuración del proceso de construcción así como con la vista del catálogo de clasificaciones.

En el primero se consigue configurar un servidor en Internet con un servicio de Amazon llamado EC2 [Amazon Elastic Compute Cloud (Amazon EC2) n.d.] para que se pueda acceder desde Internet al informe de calidad del código del proyecto, a la información sobre los últimos procesos de construcción ejecutados con Hudson. Además se configura el proyecto para que genere un repositorio P2 al construirse. Al hacer disponible este repositorio desde Internet con el repositorio de Amazon ya citado, se alcanza la meta de habilitar al usuario instalar el plugin desde Internet sin necesidad de abandonar el entorno del IDE.

En el segundo se añade la posibilidad de agregar palabras clave a una refactorización para facilitar su búsqueda y clasificación en el catálogo. Esto supone modificaciones en el asistente de creación de refactorizaciones y en el XML de las refactorizaciones. Además se

agregan a los ficheros XML de las refactorizaciones el ámbito al que pertenecen como una categoría. Con todo el modelo del dominio del catálogo ya implementado, se crea una versión de línea de comandos del catálogo para pruebas. A ésta le sigue la primera versión de interfaz gráfica dentro de Eclipse.

3.4.13. Iteración 4

Se implementan nuevas características importantes dentro de la vista del catálogo entre ellas se encuentra el filtrado de refactorizaciones desde la interfaz, nuevas pestañas con información adicional para la refactorización seleccionada y otra pestaña más para mostrar los ejemplos de las refactorizaciones.

Por otro lado se empieza a trabajar en la ayuda a la creación de refactorizaciones. Se crea un prototipo con las modificaciones al asistente necesarias y se corrige un problema en el plugin que hacía que no se mostrara correctamente la documentación de las entradas de las refactorizaciones en dicho asistente.

Con la posibilidad de instalar el plugin desde Internet se considera la necesidad de ralentizar la incorporación de características al proyecto para dedicar ciertos esfuerzos a dotar de estabilidad y liberar de fallos al plugin. Es por eso que ya desde esta fase tan temprana empiezan a incorporarse tareas de cambios internos en el diseño del proyecto y de corrección de errores. Debido a las dimensiones del proyecto y al gran número de características que incorpora este tipo de tareas se encontrarán a lo largo del resto del plan del proyecto.

3.4.14. Iteración 5

Las dos tareas principales de esta fase están relacionadas con el apartado de facilitar al usuario la creación de refactorizaciones. En esta dirección se desarrolla un completo rediseño de la búsqueda para las entradas, precondiciones, acciones y postcondiciones que permite incorporar todas las características de un motor avanzado de búsquedas gracias a la biblioteca Lucene [Apache Lucene n.d.]. También se consigue obtener la metainformación de los elementos anteriormente citados a partir de su documentación en formato JavaDoc.

Se comienza con el estudio de la línea de trabajo para resolver el problema referente al versionado de refactorizaciones. Se realizó un debate sobre las opciones más interesantes para afrontar el problema y una vez se tomó la decisión se creó un prototipo sobre dicha

solución.

Además se incorporó el firmado del plugin al proceso de construcción del proyecto, se solucionaron ciertos errores encontrados en la interfaz del asistente de creación/ediación de refactorizaciones y se incorporó una cambio interno importante en la estructura del código de las refactorizaciones pasando de utilizar un mapa para representar las entradas de las refactorizaciones a un objeto con atributos y métodos. Este cambio facilitó posteriores desarrollos por ejemplo en la línea de trabajo de la mejora del asistente de creación de refactorizaciones.

3.4.15. Iteración 6

En esta fase se llevan a cabo una serie de cambios internos de gran calado.

En primer lugar se modifica la estructura de las refactorizaciones y se pasa de tener los parámetros ambiguos definidos como un objeto difícil de comprender (un array de mapas de listas) por un modelo mucho más sencillo en el que los parámetros ambiguos son entradas y pertenecen a las precondiciones, acciones y postcondiciones.

En segundo lugar y más importante se concluye con una tarea en la que se había comenzado a trabajar en iteraciones anteriores. Esta tarea consiste en sustituir el uso en múltiples puntos del proyecto de modificaciones directas sobre los XML de definición de las refactorizaciones, por un contenedor general encargado de realizar todas las modificaciones relativas a éstas de forma centralizada.

El cambio anterior aunque es costoso nos va a facilitar notablemente implementar el versionado de las refactorizaciones en esta misma fase.

3.4.16. Iteración 7

Por problemas con el servidor web de Amazon se toma un enfoque diferente para la descarga del plugin, creando una cuenta en el servicio gratuito Google code desde la que se puede descargar el plugin. Esta modificación tiene como ventajas que la URL de instalación del plugin se mantiene constante (algo que con el servicio de Amazon no pasaba) y que la instalación es mucho más rápida y el servidor más estable.

Se lleva a cabo la tarea de versionado de clasificaciones que presentaba los mismos problemas iniciales que en el caso de las clasificaciones. Además se incorporan ciertas

mejoras al editor de clasificaciones.

Finalmente es esta fase se soluciona un error que provocaba que los controles se descuadraran en el primer paso del asistente de creación de refactorizaciones.

3.4.17. Documentación

A pesar de que la documentación se había ido realizando de forma simultánea al desarrollo del proyecto, se ha dedicado la fase final del proyecto a ir cerrando los apartados, incorporar ciertas secciones que por sus características no se podían definir hasta que el proyecto estuviera terminado y a revisar lo redactado para corregir despistes.

4. ESTUDIO DE VIABILIDAD

Un proyecto necesita de un estudio de viabilidad para conocer el dinero y tiempo que será necesario invertir y decidir si resulta rentable llevarlo a cabo.

Debemos señalar que nuestro proyecto tiene una finalidad investigadora y además está encaminado a la obtención del título de Ingeniero en Informática, aunque realizaremos el estudio como si tuviera fines comerciales.

Los estudios de viabilidad pueden ser de distintas clases:

Viabilidad técnica: se estudia la funcionalidad de un proyecto y las restricciones que pueden afectarle.

Viabilidad legal: contempla los posibles obstáculos legales.

Viabilidad económica: un análisis coste-beneficio nos indicará si el proyecto es o no rentable. Se va a dividir el estudio en tres puntos. Para empezar se estiman los costes que va a producir la creación del proyecto. Posteriormente se realiza el estudio de mercado que permite ver las posibilidades de la aplicación creada una vez a la venta. Para finalizar se exponen las decisiones tomadas en cuanto al precio del producto y se calcula el punto a partir del cual se obtienen beneficios de su venta.

Viabilidad temporal: sirve para planificar el coste temporal que tendrá un proyecto software.

4.1. Viabilidad técnica

Este es un proyecto de una elevada complejidad técnica. El hecho de que los alumnos no tuvieran ninguna experiencia previa acerca de la programación de plugins para Eclipse ha constituido el factor principal de riesgo. Esto es debido a que el entorno es ciertamente complejo por todas las características específicas que requiere, como por ejemplo el uso de *OSGI* o la dificultad de su biblioteca gráfica específica *SWT*. Además de que la disponibilidad de documentación es muy escasa.

Por otra parte el hecho de que fuera la segunda continuación de un proyecto inicial ha sido otro factor muy a tener en cuenta. Los alumnos han necesitado un comprensión de la aplicación en su totalidad y han tenido que corregir errores o introducir mejoras sobre funcionalidad previa como añadido a la funcionalidad nueva a implementar. Ha sido pues, un proyecto en el que ha sido necesario mantener las características de las versiones anteriores además de implementar nuevas funciones, con toda la carga de trabajo que eso supone.

Otros riesgos de tipo técnico existentes han surgido de la dificultad de implementar nuevas características mediante herramientas con las que los desarrolladores no estaban familiarizados. Ejemplos de esto han sido la utilización de Maven para el proceso de construcción, la incorporación de pruebas de interfaz gráfica, el control de calidad del código del proyecto o la posibilidad de instalar el plugin desde Internet por citar algunos.

4.2. Viabilidad legal

La viabilidad legal se refiere a la necesidad de determinar la inexistencia de posibles obstáculos legales para la distribución, instalación y ejecución normal del proyecto. Después del estudio de las posibles licencias a elegir se ha decidido distribuir el producto bajo un tipo de licencia *GPL*.

Además con finalidad informativa, en la siguiente tabla se muestran las licencias de los productos software usados en el desarrollo de la aplicación.

Software	Descripción	Licencia
Microsoft Windows XP Professional	Sistema operativo	Comercial
Ubuntu 10.04	Sistema operativo	Código Libre
Oracle Java JDK	Kit de desarrollo	Oracle License
Eclipse SDK Version 3.6	Entorno de desarrollo	Eclipse Public License Version (EPL)
OpenOffice 3.3	Suite de ofimática	Oracle License
Microsof Project 2007	Planificación del proyecto	Comercial
Framework de pruebas JUnit	Biblioteca de pruebas unitarias	Código Libre
Apache Maven	Herramienta de automatización del proceso de construcción	Apache Software License (libre)
Sonar	Herramienta de control de calidad del código	GPL
Jing	Capturas de pantalla	Gratuita
Apache Lucene	Biblioteca de búsqueda e indexación	Apache Software License (libre)
Balsamiq Mockup	Prototipos de interfaz gráfica	Comercial
Git	Control de versiones	GPL
Egit	Plugin de Git para Eclipse	Eclipse Public License Version (EPL)
Fogbugz	Herramienta de gestión de tareas	Comercial
SWTBot	Biblioteca para pruebas de interfaz gráfica	Eclipse Public License Version (EPL)
Hudson	Servidor de integración continua	MIT License (libre)
JaCoCo	Informes de cobertura del código de los tests	Eclipse Public License (EPL)
tycho	Plugin de Maven para construir plugins de Eclipse	Eclipse Public License (EPL)
Astah Community	Análisis y diseño UML	Gratuita

Tabla 8: Relación de licencias de las herramientas utilizadas

Como hemos comentado anteriormente se ha decidido distribuir el plugin bajo un tipo de licencia *GPL*, que a continuación va a ser explicada.

La licencia pública general de GNU o más conocida por su nombre en inglés **GNU General Public License** (*GNU GPL*), es una licencia que ha sido creada por la Free Software Foundation.

Es un tipo de licencia sobre la propiedad intelectual en la cual únicamente se exige que aquellos desarrollos hechos con material que use la licencia *GPL*, sean a su vez liberados bajo una licencia *GPL*. Está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito u objetivo es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Bajo esta filosofía, la licencia *GPL* garantiza a los usuarios de un programa software los derechos de la definición de software libre, asegurando que las libertades se conservan y preservan, incluso cuando el trabajo es modificado o ampliado.

Cabe destacar que es necesaria la compartición del código desarrollado, puesto que permite ayudar a otros desarrolladores compartiendo la resolución de problemas ya resueltos con anterioridad, lo cual permite intercambiar los avances en esta ciencia y generar una evolución más rápida de la misma.

La licencia *GNU GPL* correspondiente a la aplicación se adjunta en el CD dentro de la carpeta `Ejecutable` donde también se encuentra el plugin instalable, y también dentro del directorio con el proyecto de desarrollo del plugin en Eclipse.

4.3. Viabilidad económica

Es necesaria la realización de un estudio de viabilidad económica previo al desarrollo del proyecto, para poder estimar los recursos que van a ser utilizados y la cantidad de beneficios que éste generará.

Este estudio consiste en una evaluación de los costes de desarrollo frente a los beneficios del producto final desarrollado. La parte más importante del estudio de la viabilidad se corresponde con el análisis coste-beneficio.

4.3.1. Análisis de costes

Se debe realizar un análisis completo de todos los costes derivados del desarrollo del proyecto, estos son: costes de *personal*, de recursos *hardware*, de licencias *software* y de otros conceptos que se detallarán convenientemente.

Costes de personal

Para la realización del presente proyecto se necesitan dos desarrolladores noveles que trabajarán durante seis meses con una jornada de siete horas diarias. El salario de los trabajadores, basado en el salario medio de un programador júnior, es 10€/hora. Se debe de tener en cuenta que un mes tiene 20 días laborables.

$$10\text{€}/\text{hora} \times 7\text{horas/día} \times 20\text{días/mes} \times 6\text{meses} \times 2 \text{ sueldos} = \mathbf{16.800 \text{ €}}$$

Para realizar el proyecto se ha contado con la ayuda del tutor, las horas dedicadas por el tutor también deben ser incluidas como costes de personal. Se tiene en cuenta que se han mantenido quince reuniones con el tutor y se calcula que el promedio de duración de cada una ha sido de una hora y media. El sueldo del tutor por tratarse de un trabajador cualificado es muy superior y se ha estimado en 40 €/hora.

$$40\text{€}/\text{hora} \times 1,5\text{horas/reunión} \times 15 \text{ reuniones} = \mathbf{900 \text{ €}}$$

Costes de seguridad social

A los costes de personal hay que sumar los de seguridad social y otros impuestos añadidos que paga el empresario por cada empleado.

Los porcentajes a pagar de cada uno de estos impuestos son establecidos cada año en base a los presupuestos del estado. Para obtener los porcentajes relativos al año 2011 se ha recurrido a la página web de la seguridad social (<http://www.seg-social.es/>) [Seguridad Social n.d.]. Los valores aplicables al proyecto obtenidos son:

- Contingencias comunes: 23,6% del sueldo del trabajador.
- Impuesto de desempleo: 5,50% del sueldo del trabajador.
- Fondo de garantía social: 0,2% del sueldo del trabajador.
- Formación profesional: 0,6 % del sueldo del trabajador.

Estos impuestos deben aplicarse a la totalidad de los costes de personal, es decir, a la suma de los sueldos del tutor y de los desarrolladores.

$$(16.800+900)\text{€} \times 0,299 = \mathbf{5.292,30 \text{ €}}$$

Amortización hardware

En este apartado se incluyen los costes por amortización del hardware utilizado en el desarrollo del proyecto. Se necesitan dos ordenadores personales para el desarrollo.

El precio de adquisición de los equipos está estimado en 600€ cada uno.

Estos ordenadores se suponen que serán empleados en posteriores desarrollos. La inversión realizada en hardware deberá ser amortizada en un plazo de 3 años (36 meses), basándose en la vida útil de las máquinas antes de quedar obsoletas. Aunque el impuesto de sociedades propone como período de amortización entre 4 y 6 años, se han tomado 3 años porque se considera una estimación más real de la vida útil del hardware.

El gasto de amortización del hardware es el siguiente.

$$(6\text{meses}/36\text{meses}) \times 600\text{€} \times 2 \text{ equipos} = \mathbf{200 \text{ €}}$$

Amortización software

En este apartado se incluyen los costes ocasionados por las licencias de software que se han tenido que adquirir.

Los sistemas operativos utilizados en el desarrollo del proyecto han sido Microsoft Windows XP Professional con Service Pack 2, cuyo precio es de 250€ y la distribución Linux Ubuntu en su versión 10,04 (Lucid Lynx). La licencia de este último es de tipo *GNU GPL* [The GNU General Public License v3.0 - GNU Project - Free Software Foundation (FSF) n.d.] por lo que su uso es gratuito. Para la planificación del proyecto se ha utilizado la herramienta Microsoft Project 2007 cuyo coste es de 500€.

Para la gestión de tareas durante el proyecto se ha utilizado la web FogBugz. El coste de esta herramienta es de 25€ por usuario y por mes. Dado que la duración del proyecto ha sido de seis meses el coste de esta herramienta hubiera sido de 300€ . Este coste se va a tener en cuenta a la hora de la amortización del software pero hay que aclarar que los desarrolladores del proyecto no han necesitado incurrir en él debido a que existe una licencia gratuita para proyectos no comerciales, con un máximo de dos usuarios.

Durante el proyecto se ha utilizado la herramienta Balsamiq Mockup para el desarrollo de prototipos de interfaz gráfica. El coste de dicha herramienta es de 79€ por

usuario, con lo que el coste total para los desarrolladores se eleva a 158€. Tampoco ha sido necesario sufragar el coste de esta herramienta debido a que dispone de una versión simplificada, la cual es gratuita y se puede utilizar para la creación de pequeños prototipos.

Se ha empleado como herramienta ofimática OpenOffice. Las fases de análisis y diseño del proyecto se han realizado mediante la herramienta Astah versión *Community*, de uso libre. El entorno de desarrollo utilizado es Eclipse el cual es gratuito, al igual que las librerías de Java utilizadas y otras.

La utilización del servidor de control de versiones ha sido proporcionada por Github, que ofrece este servicio de forma gratuita para proyectos con licencia de código abierto. Lo mismo ocurre para el servidor que permite la instalación del plugin desde Internet con Google Code.

A continuación se adjuntan los costes de las herramientas no gratuitas utilizadas durante desarrollo del proyecto. Para una descripción de las funcionalidades y características de ellas se remite al apartado de *Técnicas y Herramientas de la Memoria*.

Software	Licencia
Microsoft Windows XP Professional	250 €
Microsof Project 2007	500 €
Fogbugz	300 €
Balsamiq Mockup	158 €
Total Software	1.208 €

Tabla 9: Coste de las herramientas utilizadas

El periodo de amortización del software se establece en tres años (36 meses), a partir de los cuales se considera que las aplicaciones pierden su valor. Según este criterio el gasto de amortización es el siguiente:

$$(6\text{meses}/36\text{meses}) \times 1.208\text{€} = \mathbf{201,33\text{ €}}$$

Otros gastos

Se trata de otros gastos que son necesarios, como son el coste del material de oficina empleado cuyo coste aproximado sería de unos 200 €. Se incluye en este apartado el coste porcentual de los suministros (calefacción, energía eléctrica, teléfono, conexión a Internet).

También se debe tener en cuenta el coste del alquiler de las instalaciones de trabajo, puesto que es un recurso necesario para la realización del proyecto. El coste del alquiler se ha calculado en 300€/mes, dado que el período de desarrollo ha sido de seis meses el coste derivado del alquiler de las instalaciones asciende a:

$$300\text{€}/\text{mes} \times 6\text{meses} = 1.800 \text{ €}$$

La suma de los conceptos anteriores, es decir, el total de otros gastos es:

$$200\text{€} + 1.800\text{€} = \mathbf{2.000 \text{ €}}$$

El coste total del proyecto se especifica a continuación.

Concepto	Cantidad
Coste personal	17.700,00 €
Coste Seguridad Social	5.292,30 €
Coste hardware	200,00 €
Coste software	201,33 €
Otros gastos	2.000,00 €
Coste Total	25.393,63 €

Tabla 10: Costes del proyecto

4.3.2. Análisis coste-beneficio

Este apartado esta dedicado al análisis de coste-beneficio, en él se va a realizar un estudio de la competencia así como el estudio de mercado que permite ver las posibilidades de la aplicación creada una vez a la venta. Para finalizar se van a exponer los beneficios económicos y no económicos, en el primero se incluirán las decisiones tomadas en cuanto al precio del producto y se calculará el punto a partir del cual se obtienen beneficios de su venta.

Estudio de la competencia

De entre las alternativas ya estudiadas en la *Memoria* se van a considerar las dos que ofrecen una funcionalidad más similar: Refactory y RefactoringNG. El resto de alternativas no han sido consideradas ya que o no permiten definir refactorizaciones propias o es muy compleja su definición lo que descarta la utilización de esta funcionalidad para la mayoría de los usuarios, por lo tanto no compiten en sentido directo con el plugin de refactorización.

Refactory es una herramienta muy flexible, cuyo punto fuerte es que permite definir refactorizaciones que son totalmente independientes del lenguaje con el que se trabaja. Esto la hace una herramienta de gran valor especialmente para aquellos que utilizan el proceso de desarrollo basado en modelos y definen sus propios lenguajes específicos para su dominio (*DSL*). Sin embargo, por su planteamiento tan general y tan independiente del lenguaje este plugin no va a permitir definir refactorizaciones avanzadas para Java. Además tampoco permite definir refactorizaciones sobre fragmentos de código como por ejemplo *Extract Method*. En definitiva, a pesar de ser un plugin muy interesante Refactory posee desventajas notables respecto al plugin de refactorización cuando se quiere utilizar para el lenguaje de programación Java.

RefactoringNG por su parte sí permite definir refactorizaciones que utilizan todas las propiedades de los *árboles AST* de Java, al contrario de lo que sucedía con Refactory. La desventaja de RefactoringNG es que carece de la semántica necesaria para poder definir precondiciones y poscondiciones o para tener en cuenta las relaciones entre las clases de un modelo. Es por ello que RefactoringNG tampoco supone una competencia directa para el plugin de refactorización.

Análisis del mercado

El perfil del usuario que podría estar interesado en nuestro plugin sería el siguiente:

- Utiliza Eclipse como entorno de programación.
- Aprecia el valor que aportan las refactorizaciones al desarrollo de software.
- Java es su lenguaje de programación.
- Para el uso de ciertas características del plugin de refactorización el usuario debe poseer unos conocimientos de Java y de Eclipse avanzados.
- Presta atención al buen diseño y a la mejora constante de su código.

Beneficios no económicos

Gracias al plugin los desarrolladores que utilizan Eclipse podrán beneficiarse de la siguiente funcionalidad:

- Una vista que permite el estudio de un catálogo amplio de refactorizaciones con abundante información descriptiva y recursos adicionales como imágenes y ejemplos de código.
- La posibilidad de organizar la forma en que se visualizan las refactorizaciones, modificando la clasificación seleccionada y filtrando por una serie de criterios.
- La opción de poder definir nuevas clasificaciones propias y de asignar nuevas categorías a las refactorizaciones.
- La posibilidad de obtener refactorizaciones nuevas por cada actualización del plugin, sin necesidad de eliminar las refactorizaciones que el usuario había definido personalmente.
- Ayuda a la hora de crear las refactorizaciones, con mejoras en el asistente tales como búsqueda avanzada de elementos por nombre y por descripción o relación de las refactorizaciones en las que los elementos participan.
- Posibilidad de instalar el plugin desde Internet y obtener actualizaciones del mismo de forma automática y sin necesidad de abandonar Eclipse.

- Todo esto como añadido a las características de las que el plugin ya disponía como son: la importación y exportación de refactorizaciones, de planes de refactorizaciones o el historial de refactorizaciones aplicadas.

En lo referente al equipo de desarrollo, es fácil ver el enorme beneficio que va a adquirir durante la ejecución del proyecto. Adquirirán el conocimiento necesario para el trabajo en grupo, desarrollarán su liderazgo, potenciarán la toma de decisiones y obtendrán mucha experiencia a nivel técnico; desde programación en Java pasando por el manejo de entornos *OSGI*, la programación de plugins para *Eclipse*, los procesos de construcción basados en *Maven* y el trabajo con ficheros *XML*, entre otras cosas. Con todo ello, se puede deducir que desde un punto de vista no económico el proyecto reporta grandes beneficios.

Beneficios económicos

Puesto que la herramienta desarrollada no está pensada como aplicación comercial, resulta difícil estimar el precio para una licencia. Lo que se va a hacer es utilizar como punto de partida el precio de *IntelliJIDEA*, una solución comercial que proporciona alguna de las características del plugin de refactorización. Dicha herramienta no ha sido considerada como alternativa en la sección de *Comparativa entre alternativas de la Memoria* porque es un entorno integrado de desarrollo y no se centra exclusivamente en las refactorizaciones, además de que no permite definir refactorizaciones propias. Los precios de los distintos tipos de licencias comerciales de *IntelliJIDEA* se reflejan en la siguiente tabla:

Tipo de licencia	Precio
Commercial License	420 €
Personal License	168 €
Academic License	84 €

Tabla 11: Precios de las licencias comerciales de IntelliJIDEA

Se consideró que un paquete que pudiese incluir un entorno de desarrollo como lo es *Eclipse* junto con el plugin de refactorización contaría con unas características agregadas con las que *IntelliJIDEA* no cuenta, por ejemplo la posibilidad de definir refactorizaciones propias o de clasificar las refactorizaciones y visualizarlas en un catálogo, por lo tanto sería interesante para los desarrolladores. Teniendo en cuenta esta consideración, pero también el hecho de que lo que se está ofertando a los clientes no es un entorno de desarrollo integrado sino un extensión del mismo, el precio mínimo que se estableció para la licencia comercial fue de 100 euros.

Presentamos a continuación una tabla comparativa entre el número de licencias que deberán venderse para amortizar el coste estimado partiendo de los 100 euros que se han establecido como precio mínimo. Con ventas mayores al número de licencias indicado en la tabla se obtendrá beneficio.

Precio(€)	Nº licencias
100	≈ 254
120	≈ 212
140	≈ 182
180	≈ 141

Tabla 12: Relación entre el precio y el número de licencias de la aplicación

Definitivamente, teniendo en cuenta las características distintivas existentes respecto de IntelliJIDEA y el número de licencias necesarias para empezar a rentabilizar la inversión se decidió utilizar como precio del producto los 120€.

Con dicho precio sería necesario vender unas 212 licencias para recuperar la inversión y empezar a obtener beneficios:

$$25.393,63 \text{ € de coste total} / 120 \text{ € por licencia} = \underline{\underline{212 \text{ licencias}}}$$

Se consideró que era factible vender ese número de licencias teniendo en cuenta que el precio de la aplicación no es excesivamente alto y que la funcionalidad del plugin es atractiva y puede interesar a una amplia comunidad de desarrolladores que programan en Java utilizando Eclipse. Este objetivo se podría ver facilitado mediante una campaña de promoción del producto en la que se ofreciera a los usuarios una versión con características reducidas instalables sencillamente a través de Internet. Una vez que los usuarios hubieran podido probar dicha versión tendrían la posibilidad de acceder a la versión completa mediante el pago de la licencia. Además otra estrategia a seguir para rentabilizarlo sería la prestación de servicios de mantenimiento y mejora, la creación de módulos o paquetes específicos bajo demanda, etc.

Análisis basado en el V.A.N (Valor actual neto)

Consiste en actualizar a valor presente los flujos de caja futuros que se prevé va a generar el proyecto, descontados a un cierto tipo de interés (la tasa de descuento), y compararlos con el importe inicial de la inversión. Si el VAN es mayor que cero se considera

que el proyecto es rentable y si es menor se considera que no lo es. Desde el punto de vista de un inversor a la hora de escoger entre dos proyectos, elegirá aquel que tenga el mayor VAN.

La fórmula para calcular el VAN es la siguiente:

$$\text{VAN} = \sum (V_t / (1 + k)^t) - I_0$$

donde;

V_t representa los flujos de caja en cada periodo **t**.

I₀ es el valor del desembolso inicial de la inversión.

n es el número de períodos considerado.

k es la tasa de descuento.

Como ejemplo de tasas de descuento (o de corte) que se suelen utilizar indicamos las siguientes:

- Tasa de descuento ajustada al riesgo = Interés que se puede obtener del dinero en inversiones sin riesgo (deuda pública) + prima de riesgo.
- Coste medio ponderado del capital empleado en el proyecto.
- Coste de la deuda, si el proyecto se financia en su totalidad mediante préstamo o capital ajeno.
- Coste medio ponderado del capital empleado por la empresa.
- Coste de oportunidad del dinero, entendiendo como tal el mejor uso alternativo, incluyendo todas sus posibles utilizaciones.

Otra opción es definir una tasa de descuento con el rendimiento que se espera de la inversión para considerarla rentable. Para definir esta tasa se suele tener en cuenta que se obtenga un beneficio superior al que ofrecería el invertir el desembolso inicial a plazo fijo o en bonos del tesoro. En nuestro caso se han tenido en cuenta las estadísticas publicadas en [Banco de España - Tipos de interés n.d.] y en base a ello se han tomado tres tasas de descuento distintas para evaluar la rentabilidad del proyecto: 4,5%, 5,5% y 7%.

La tabla siguiente muestra los valores que se han utilizado para el cálculo del VAN. Cada fila corresponde con el período correspondiente a un año y a la derecha se muestran el número de licencias que se prevé vender a lo largo de ese año y los beneficios que se derivarían de esas ventas. En el primer periodo se refleja el desembolso inicial del proyecto.

Período	Nº licencias	Flujos de caja
0	-	-25.393,63
1	50	6.000
2	125	15.000
3	50	6.000
4	25	3.000

Tabla 13: Valores utilizados para el cálculo del VAN

En el primer año el producto no sería muy conocido por lo que se conseguirían ventas moderadas. En el segundo año con un producto más consolidado en el mercado y más popular las ventas se incrementarían notablemente para luego ir descendiendo en años posteriores porque la evolución de otras herramientas iría convirtiendo el plugin en un producto obsoleto, a menos que este siguiera evolucionando.

El resultado de aplicar la fórmula del VAN sobre las tres tasas de descuento establecidas anteriormente es la siguiente:

$$k = 4,5\% \Rightarrow VAN = \sum (V_t / (i + k)^t) - I_0 = 1.777,43\text{€}$$

$$k = 5\% \Rightarrow VAN = \sum (V_t / (i + k)^t) - I_0 = 1.502,12\text{€}$$

$$k = 7\% \Rightarrow VAN = \sum (V_t / (i + k)^t) - I_0 = 469,07\text{€}$$

Como se puede observar el VAN indica que el proyecto sería rentable para cualquiera de las tres tasas de descuento aplicadas. Incluso cuando se aplica una tasa de descuento exigente del 7% el VAN continua devolviendo un valor positivo, lo que representa un indicador muy favorable.

La rentabilidad del proyecto se debería principalmente a que ofrece una funcionalidad atractiva que puede interesar a una comunidad amplia de desarrolladores. Sin embargo, es necesario puntualizar que el VAN es un estimador impreciso en el caso de proyectos tecnológicos debido a la rápida obsolescencia de los mismos. Si se quisiera hacer una estimación más precisa habría que restar a los beneficios de cada periodo una cantidad

asociada al coste necesario para realizar el mantenimiento de la herramienta.

4.4. Viabilidad temporal

El estudio de la viabilidad temporal sirve para planificar el coste temporal que tendrá un proyecto software. Para ello se tienen en cuenta diversos factores que intervienen en el desarrollo del mismo.

Utilizando los casos de uso, elementos que ya se tienen establecidos por medio de los requisitos exigibles para la elaboración del proyecto, y a través de una serie de cálculos se llegará a una estimación del tiempo necesario para desarrollar el proyecto y comprobar si es viable o no.

4.4.1. Estimación de Casos de Uso

En este apartado vamos a utilizar técnicas de estimación del trabajo basadas en los casos de uso.

Peso de los actores

Se asigna un factor clasificándolos si son actores simples (1), promedios (2) o complejos (3). [Pressman 2006]

En nuestro caso tenemos un único actor:

Usuario: interactúa con una interfaz gráfica. Factor 3.

El peso total de los actores es la suma de los factores de cada actor:

$$\textbf{Peso de los actores} = 3 \text{ (usuario)} = \mathbf{3}$$

Peso de los casos de uso

Sólo se consideran los casos de uso que interactúan con actores, su clasificación depende del número de transacciones del caso de uso, pudiendo ser:

- Simples: representan 3 transacciones o menos (factor 5)
- Promedio: representan de 4 a 7 transacciones (factor 10)
- Complejo: representan más de 7 transacciones (factor 15)

Veamos una tabla con los factores de los casos de uso definidos en el análisis:

Caso de uso	Nº transacciones	Factor
Visualizar refactorizaciones según clasificación	3	5
Refrescar visualización de refactorizaciones	2	5
Añadir filtro de refactorizaciones	6	10
Seleccionar opción aplicar filtro	3	5
Eliminar filtro de refactorizaciones	3	5
Eliminar todos los filtros	2	5
Seleccionar opción ver refactorizaciones filtradas	2	5
Visualizar detalle de refactorización	3	5
Añadir clasificación	3	5
Editar clasificación	4	10
Eliminar clasificación	3	5
Añadir categoría a una clasificación	3	5
Renombrar categoria de una clasificación	4	10
Eliminar categoria de una clasificación	5	10
Mostrar resumen de elemento seleccionado	3	5
Realizar búsqueda de elemento	3	5

Tabla 14: Peso de los casos de uso

El peso total de los casos de uso es la suma de los factores de cada caso de uso:

$$\text{Peso de los casos de uso} = \sum F_i = 100$$

UUCP: Puntos de Casos de Uso sin Ajustar.

Se calcula sumando los pesos de los actores y de los casos de uso.

$$UUCP = \text{Peso actores} + \text{Peso de casos de uso}$$

$$\text{UUCP} = 3 + 100 = 103$$

Peso de los factores técnicos: se calcula con el TFC Factor Técnico de Complejidad

$$\text{TFC} = 0,6 + (0,01 \times \sum (F_i \times \text{Peso}_i))$$

;donde:

Peso es un número de 1 al 5 que se asigna según su importancia a cada uno de los factores de la tabla siguiente:

Factor	F_i	Peso	$F_i \times \text{Peso}$
Sistema Distribuido	2	1	2
Tiempos de respuesta críticos	1	3	3
En Línea	1	1	1
Procesos internos complejos	1	5	5
Código reutilizable	1	4	4
Fácil de instalar	0,5	5	2,5
Fácil de utilizar	0,5	5	2,5
Portable	2	3	6
Fácil de modificar	1	5	5
Concurrencia	1	1	1
Incluye características de seguridad	1	1	1
Acceso a software creado por otras compañías	1	4	4
Incluye facilidades de aprendizaje para el usuario	1	4	4

Tabla 15: Peso de los factores técnicos

$$\sum (F_i \times \text{Peso}_i) = 41$$

$$\text{TFC} = 0,6 + (0,01 \times 41) = 1,01$$

EF: Factor de Entorno. Indica el conocimiento que tiene el programador del tipo de aplicaciones que desarrolla. Para calcularlo se le asignan unos pesos a una serie de factores, desde 0 (nada de conocimiento previo) hasta 5 (experto).

$$\text{EF} = 1,4 + (-0,03 \times \sum (F_i \times \text{Peso}))$$

Factor	F_i	Peso	$F_i \times Peso$
Familiarizado con ciclo de vida	1,5	3	4,5
Experiencia con este tipo de aplicaciones	0,5	2	1
Experiencia en orientación a objetos	1	4	4
Capacidad de liderazgo del analista	0,5	3	1,5
Motivación	1	3	3
Requisitos estables	2	4	8
Trabajadores a tiempo parcial	-1	3	-3
Lenguaje de programación difícil	-1	3	-3

Tabla 16: Peso de los factores de entorno

$$\sum (F_i \times Peso_i) = 16$$

$$EF = 1,4 + (-0,03 \times 16) = 0,92$$

UCP: puntos de casos de uso.

$$UCP = UUCP \times TFC \times EF$$

$$UCP = 103 \times 1,01 \times 0,92 = 95,7076$$

Con todos los datos calculados ya podemos obtener las horas hombre necesarias.

Tenemos que fijarnos en los factores de puntuación del EF que son inferiores al nivel promedio.

$$\text{Horas persona} = 20 \times UCP = 20 \times 95,7076 = 1.914,152$$

Si en el proyecto trabajan 2 personas, durante 7 horas diarias y 20 días al mes:

$$\begin{aligned} \text{Tiempo} &= 1914,152 \text{ Horas Persona}/2 Personas \times 1 \text{ Día}/7 \text{ Horas} \times 1 \text{ Mes}/20 \text{ Días} = \\ &6,8 \text{ meses} \end{aligned}$$

Nuestro proyecto ha sido desarrollado en un periodo total de 6 meses pero durante estos meses han existido etapas en las que se ha trabajado de una forma más intensa superado los 20dias/mes de trabajo que han sido marcados como referencia, debido a necesidades del proyecto y también por la propia motivación del equipo de trabajo, de ahí se justifica que esta estimación obtenida para el tiempo necesario para el desarrollo del proyecto sea algo superior, en concreto algo más de 6 meses y medio.