INDICE

1. INTRODUCCIÓN

Con esta memoria se pretende recoger el trabajo realizado tanto a nivel teórico como nivel práctico de estos últimos meses, en el proyecto *Sistema de ayuda al diseño de experimentos: aplicación a la selección de atributos con SVMs*.

La lectura de este documento permite conocer con mayor profundidad cómo se ha construido el sistema, con qué finalidad se ha construido y qué representa el sistema. Además queda constancia de cuáles son las posibilidades del sistema y cual era la idea de partida cómo se ha construido y que resultado ha dado. Por último se puede ver qué posibilidades tiene el sistema en cuánto a posibles ampliaciones o mejoras.

1.1 Agradecimientos

El acabar este proyecto ha supuesto un gran reto, tanto a nivel personal, cómo a nivel académico, cómo a nivel profesional.

Quisiera agradecer en primer lugar la dedicación paciencia de Enrique Romero, que después de tanto tiempo en la cresta de esta ola ha sabido orientarme y aconsejarme para llegar al fin.

También agradecer la perseverancia y comprensión de mi familia y de Noe para que trabajara en este proyecto en cualquier momento posible dentro de mi apretada agenda.

A mis amigos y compañeros de trabajo que también han puesto su granito de arena y han aguantado pacientemente mis sesiones explicativas de lo que representaba este proyecto.

A todos aquellos que me dejo en el tintero, no otra falta que mi memoria.

1.2 Origen del proyecto

Este proyecto surge por la necesidad de establecer un cierto orden en las prácticas de investigación utilizando scripts.

1.3 Entorno del proyecto

En la realización de un proyecto hay un contexto que define las necesidades y también el camino a seguir para establecer, inicialmente, los objetivos.

La implantación de un sistema cómo el que se ha desarrollado en un entorno de investigación puede suponer un estrato más en la organización de qué código se utiliza a la hora de experimentar.

En el siguiente capítulo se introduce una descripción del contexto que se utiliza a la hora de hacer los experimentos.

1.3.1 Qué es un experimento

Un experimento es un procedimiento metódico en el que se trata de verificar si una hipótesis inicial es cierta. Para llevar a cabo este procedimiento se parte de unas variables que presumiblemente definen la causa del fenómeno o resultado que se quiere verificar.

Es importante diseñar el experimento de manera que facilite la consecución de los objetivos marcados y poder ver si se cumple la hipótesis o no, además de que permita realizar varias repeticiones para ver cómo afecta las variables en diferentes condiciones.

1.4 Breve descripción

Gexp es un sistema que permite gestionar proyectos formados por:

- Los pasos que forman el proyecto
- Los diferentes scripts que pueden formar un paso
- Los diferentes comandos que forman un script
- Las variables y cadenas de códigos que forman los diferentes comandos

Las principales funcionalidades del sistema son:

- Definir, modificar y eliminar un proyecto.
- Duplicar un proyecto
- Generar un conjunto de scripts de manera ordenada
- Copiar pasos, scripts o comandos de un proyecto a otro proyecto

- Gestionar pasos del proyecto
- Gestionar scripts del proyecto
- Gestionar comandos de los scripts

El diseño del sistema está estructurado por capas y permite incorporar funcionalidades nuevas de una manera sencilla.

1.5 Motivación personal

La principal motivación de este proyecto ha sido acabar la carrera de ingeniería que estoy cursando, aunque no ha sido el único, ni mucho menos.

Otra motivación ha sido el poder desarrollar desde origen un sistema útil, intentándome basar en muchos conceptos adquiridos en los estudios y, que por desgracia, en el mercado laboral por cuestiones de tiempo muchas veces no se pueden desarrollar.

También me ha motivado el hecho de que el entorno de programación fuera nuevo para mí, ya que toda mi experiencia se basa en entornos de Microsoft y el proyecto está basado en un entorno Linux.

Durante el desarrollo del proyecto, además, surgieron retos inesperados que también han contribuido a la consecución de este proyecto.

1.6 Organización de la memoria

Este documento pretende ser una guía útil de todo el trabajo realizado y la manera en que se ha realizado. Comprende desde la descripción del entorno hasta las fases de desarrollo, la definición de objetivos, técnicas y metodologías utilizadas e inconvenientes y soluciones encontrados a lo largo del camino.

A continuación se hace una breve descripción de la estructuración de este documento:

- **Objetivos:** Antes de realizar cualquier proyecto se marcan unos objetivos que definirán la trayectoria a seguir. Se describe el sentido de la consecución de estos objetivos.

- **Metodología:** Se explican los métodos utilizados para elaborar cada una de las fases de la construcción del sistema.

- **Análisis de requerimientos:** Se establecen formalmente las necesidades que se deducen del entorno del proyecto para marcar el punto inicial al diseño del sistema.

- **Planificación y costes:** En esta sección describe una estimación del coste y planificación del proyecto en función de los análisis de requerimientos.

- **Especificación:** Pasos realizados para especificar las necesidades del sistema, marcadas por el análisis de requerimientos.

- **Diseño:** A partir de la especificación se establece la metodología de diseño y las decisiones tomadas para conseguir los objetivos del punto anterior.

- **Implementación:** Siguiendo el contenido de las dos secciones anteriores se explica cómo y porqué se ha implementado el sistema en la manera que está hecho.

- **Conclusiones:** Al finalizar la implementación se establece si se han conseguido los sistemas y la desviación o consecución de los objetivos iniciales.

- **Manual de usuario:** En este apartado se describe cómo utilizar la funcionalidad del sistema.

- **Bibliografía:** Fuentes de información que se han utilizado tanto para realizar la documentación como para realizar el sistema.

2. OBJETIVOS

Una vez descrito el entorno de proyecto es turno de identificar y describir los objetivos que se quieren conseguir con el proceso de desarrollo del sistema de este proyecto.

Una vez realizada esta tarea se pueden describir, entonces, el análisis de requerimientos, el análisis funcional, el diseño y la implementación de este sistema.

2.1 Descripción de los objetivos

El principal objetivo del proyecto es minimizar el esfuerzo de construir y gestionar scripts para realizar experimentos.

Hasta ahora, el construir un experimento supone varias etapas. Por un lado se ha de gestionar toda la información recogida de las fuentes que alimentarán el experimento.

También se ha de tener en cuenta los diferentes procesos que se utilizarán para preprocesar y tratar las diferentes fuentes del experimento, hasta aplicar el proceso principal, objeto de estudio en el experimento.

Por otro lado se debe gestionar también las salidas del experimento tras aplicarle el proceso principal.

El principal inconveniente a la hora de hacer experimentos, en el entorno descrito en el capítulo anterior, es que muchas veces se aplican procesos parecidos en cuanto a metodología, con el único cambio de variables, preprocesos o aplicaciones sobre las fuentes de datos. Es por eso que el

objetivo principal se centrará en este punto, dejando abierta la gestión de las fuentes y las salidas de los experimentos.
Otro objetivo, dependiente del definido en el anterior párrafo, será el de facilitar la portabilidad y la mantenibilidad del sistema.

Se quiere evitar el uso de sistemas gestores de bases de datos, ya que requieren una previa instalación de unos de los muchos que hay en el mercado o el acceso a un servidor que disponga de uno de ellos.

Para facilitar la gestión de todos estos procesos, generalmente representados por scripts, se implementará una herramienta visual que se encargue de manejar toda la información, en definitiva código y asociaciones entre este código de una manera intuitiva.

2.2 Alcance del sistema

Se describen a continuación en esta sección, y, después de haber establecido los objetivos y haber ordenado la información recibida, las funcionalidades del sistema a desarrollar y cuáles no se desarrollarán aunque se deje la posibilidad de hacerlo.

2.2.1 Funcionalidades acordadas

Las funcionalidades aquí descritas son las que se han establecido con el usuario del sistema:

- Gestión de diferentes experimentos, denominados a partir de ahora en el sistema como "proyectos". Se gestiona su ubicación y los diferentes pasos que lo forman.
- Con la finalidad de simplificar según que experimentos o proyectos que son parecidos a algunos realizados anteriormente, el sistema permite duplicar el entorno de esos proyectos realizados anteriormente.
- Gestión de la composición de los diferentes pasos de los proyectos y de los diferentes procesos, representados por scripts, que forman dichos pasos.
- Gestión de los diferentes scripts, descritos en el punto anterior, que representan un proceso y están formados por diferentes comandos parametrizados de código.
- Gestión de los diferentes comandos, componentes de los scripts.
- Gestión de los parámetros incluidos en los comandos.

2.2.2 Funcionalidades no acordadas

Las funcionalidades descritas en este punto son funcionalidades que a priori no se incluyen dentro del sistema. Se deja, sin embargo, la opción de poder desarrollar las siguientes funcionalidades en un futuro ó, si la planificación de tiempo, siendo muy optimista, permite su desarrollo:

- Copia de pasos, scripts o comandos de proyectos anteriores.
- Ejecución de comandos, scripts o pasos.
- Resultado de la ejecución de comandos.
- Compilación con validación de código.

2.3 Fases del proyecto

Las funcionalidades acordadas en el anterior apartado, determina la manera organizativa de acometer la planificación del desarrollo del sistema.

La idea es que para poder gestionar un experimento o proyecto, primero se debe poder gestionar todos sus componentes.

Se establecen por tanto dos fases en el desarrollo:

Fase	Funcionalidades
I	1. Gestión de los parámetros incluidos en los comandos. 2. Gestión de los diferentes comandos. 3. Gestión de los scripts 4. Gestión de los pasos.
II	1. Gestión de los proyectos. 2. Duplicación de proyectos. 3. Construcción del código y su estructura.

2.4 Objetivos personales

Muchos proyectos, además de suscitar el interés de solucionar unas necesidades o requisitos profesionales, también pretenden satisfacer unos objetivos personales de los integrantes del equipo de desarrollo.

No se puede decir que mi vida laboral profesional haya sido muy extensa, pero sí que ha aportado condicionantes a la hora de fijar los objetivos.

Los objetivos personales marcados a priori son los siguientes:

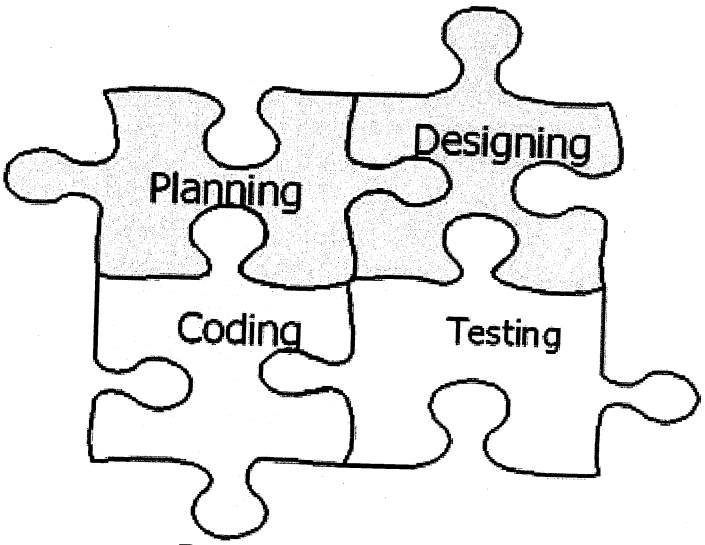
- Desarrollar un sistema que fuera fácilmente mantenible y extensible.
- Explorar el desarrollo parametrizado utilizando funciones de metalenguaje.
- Experimentar con otras herramientas y técnicas desconocidas para mí.

- Resolver un problema diferente, dentro de un ámbito diferente al de mi vida profesional.

3. METODOLOGIA

Existen muchas metodologías a la hora de desarrollar un proyecto. Este proyecto podría estar basado en una metodología tradicional, basada en el control del proceso mediante una definición rigurosa de tareas. Se ha comprobado, sin embargo, que a veces este tipo de metodología no acaba de funcionar. El control exhaustivo de un proyecto de mediana envergadura acaba mermando los recursos.

Por otro lado, en el año 2001, un grupo de expertos del software, esbozan los principios para permitir desarrollar software de una manera rápida respondiendo eficazmente a los cambios que se producen en el transcurso del desarrollo. En la reunión realizada por estos expertos aparece el "Manifiesto Ágil", y, de este manifiesto, las metodologías ágiles.



Esquema de metodología "ágil"

Este proyecto no utiliza una metodología tradicional ni tampoco una metodología ágil. Este proyecto utiliza partes de cada una en función de la necesidad establecida desde un principio y la que ha surgido en el transcurso del desarrollo.

Sobre lo que sí que se ha priorizado en la metodología usada en este proyecto son algunos de estos principios:

- Simplicidad: se ha pretendido desarrollar un diseño sencillo que permitiera en todo momento ampliar, cambiar o mantener sin gran esfuerzo el trabajo a realizar.
- Refactorización: en pro de la mantenibilidad del sistema, cada vez que se detectaba una tarea reiterativa, se refactorizaba esta tarea para simplificar el entendimiento de los procesos que usan esta tarea.
- Implicación del usuario: para que un sistema desarrollado acabe siendo útil se ha de basar en las necesidades de los usuarios. Una constante comunicación facilita esta tarea.
- Construcción en fases: se ha visto el sistema global como pequeños sistemas. De esta manera se establecen metas más fáciles de alcanzar y, además validadas.

3.1 Análisis de requerimientos

Una vez establecidos unos objetivos conviene definir qué se necesita para poder alcanzarlos. El análisis de requerimientos tiene esta finalidad.

Se distinguen tres etapas en la definición de los requerimientos:

Etapa 1

En esta etapa se ve con el usuario, que el realizar las tareas de mantenimiento de diseño de un experimento es una tarea pesada. Se establece entonces la necesidad de desarrollar una herramienta que facilite esta tarea.

Etapa 2

Después de haber establecido la necesidad de una herramienta se profundiza en el entorno en el que se desarrollan las tareas que se quieren gestionar con esta herramienta. Al profundizar se establece una primera estructuración y forma organizativa para los diferentes agentes implicados.

Etapa 3

Se concreta exactamente la estructura organizativa y que procesos se realizarán sobre ella. Se descartan opciones concretas de organización del sistema, dentro de la organización general, que son redundantes o que no aportan una utilidad explícita.

Todas estas etapas quedarán más detalladas en próximos capítulos.

3.2 Métodos de especificación, diseño e implementación

Para definir la estructuración organizativa del sistema, de la que se referenciaba en el capítulo anterior, se considera realizar una especificación y un diseño del sistema.

Como herramienta, para la especificación, se utiliza el lenguaje UML, que es útil y muy descriptivo para describir la estructura, el comportamiento y la arquitectura del sistema de información.

Al igual que para la especificación, también para el diseño se utiliza como herramienta el lenguaje UML.

La implementación del sistema vendrá condicionada por varios requisitos:

- Se pretende utilizar la herramienta desarrollada en un entorno Unix
- No se pretende tener grandes necesidades ni requisitos a nivel de software para poder utilizar la herramienta
- No se quiere obligar al usuario a utilizar un sistema gestor de bases de datos concreto
- Generalmente un sistema gestor de base de datos obliga a utilizar instrucciones para modificar los datos. Se desea modificar la información sin tener que utilizar ninguna instrucción.
- El lenguaje de programación debe ser lo suficientemente potente para poder gestionar todo el resto de requisitos.

Por los anteriores requisitos se decide escoger el lenguaje Java como plataforma principal de desarrollo y XML (e**X**tensible **M**arkup **L**anguage) metalenguaje que permite almacenar la información que generará el sistema de una manera ordenada y permite la compatibilidad entre sistemas, al igual que Java, de manera que el sistema no quedará restringido.

Además de estas dos herramientas se han usado otras para la implementación, que se explica con más detalle en los capítulos siguientes.

3.3 Documentación y memoria

Para la realización de este documento se ha utilizado un procesador de textos, Microsoft Word 2003, que tiene una potencia más que suficiente para maquetar la información que aquí se recoge.

4. ANALISIS DE REQUERIMIENTOS

El análisis de requerimientos es una de las fases más importantes dentro del desarrollo de un sistema software. Concretamente define un conjunto de restricciones positivas y negativas que el debe recoger el sistema con la finalidad de cumplir los objetivos marcados en primera instancia.

Cada una de las restricciones (tanto las positivas como las negativas) que conforman el análisis de requerimientos permiten modelar el sistema.

Se distinguen dos tipos de requerimientos:

- Funcionales: hacen referencia a los procesos que el sistema debe llevar a cabo.
- No Funcionales: hacen referencia a propiedades intrínsecas del sistema (fiabilidad, escalabilidad, portabilidad...)

A continuación se describe el proceso seguido para realizar el análisis de requerimientos.

4.1 Análisis de la organización de experimentos

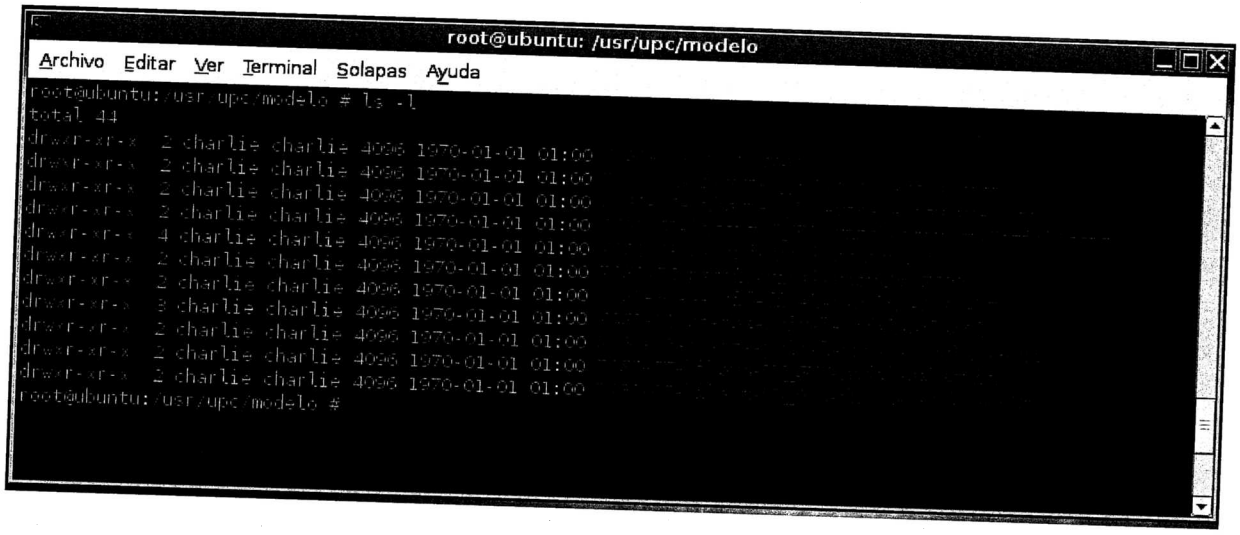
Actualmente los usuarios que realizan experimentos se basan en la hipótesis inicial que quieren comprobar y, luego en una serie de pruebas realizadas sobre un conjunto de datos con unos determinados procesos que permiten establecer unas conclusiones acerca de esa hipótesis inicial.

Los procesos mencionados suelen ser ejecutables que son llamados desde scripts. Un script es un conjunto de instrucciones, sentencias de control,

variables y demás elementos de programación almacenados en un fichero de texto y que son interpretados en la ejecución.

La estructuración de las pruebas se basa en guardar en unas carpetas las diferentes etapas de la experimentación. Cada una de estas etapas es llamada "PASO" y se refleja en el nombre de la carpeta que guarda los scripts que componen esa etapa.

Abriendo una de las carpetas que representan un "PASO" se puede encontrar, en general, scripts escritos con el shell tcshell y scripts de otros lenguajes propios de procesos que se lanzan en estos experimentos.



Se puede observar, pues, que se sigue un orden el diseño de las pruebas. A pesar de este orden hay tareas que pueden a llegar a ser tediosas.

En el momento que se quiere repetir un experimento en el que sólo se modifican unas determinadas variables (por ejemplo el path de archivos de pruebas) puede ser pesado ir script por script modificando las referencias de archivos.

Si se tienen definidos varios experimentos sobre un mismo tema puede ser realmente caótico acordarse al cabo de un tiempo que variables son las que diferían de un experimento a otro.

4.2 Ideas que permanecen

Después de observar la estructuración descrita en el apartado anterior se decide mantener una estructuración parecida, utilizando prácticamente los mismos conceptos:

- Los experimentos pasan a llamarse "Proyectos"
- Un proyecto está formado por un número determinado de pasos

- Los pasos están formados por scripts.
- Finalmente, los scripts contienen comandos que permiten definir procesos a bajo nivel.

Dentro de los scripts, los comandos utilizaran variables (de diferentes tipos) que ayudarán a parametrizar estos comandos.

4.3 Requerimientos

Tal como se habló al principio del capítulo se disponen de dos tipos de requerimientos: funcionales y no funcionales. En este apartado se realizará el análisis de ambos tipos.

4.3.1 Requerimientos funcionales

Después de realizar el análisis del entorno, los componentes y variables que intervienen en el actual sistema se pueden empezar a definir los diferentes requisitos funcionales del sistema que se va a construir. Se clasifican de la siguiente manera, con la finalidad de hacer más cómoda su lectura:

- Requerimientos de proyectos.
- Requerimientos de pasos.
- Requerimientos de scripts.
- Requerimientos de comandos.
- Requerimientos procedurales del sistema.

4.3.1.1 Requerimientos funcionales de proyectos

En la tabla siguiente se definen los requerimientos funcionales de proyectos:

Requerimientos
El sistema permite dar de alta nuevos proyectos.
El sistema permite modificar la información del proyecto.
El sistema permite especificar la ruta dónde está contenido el diccionario del proyecto.
El sistema permite modificar el diccionario asociado al proyecto.
El sistema permite asignar los pasos definidos en el diccionario al proyecto.
El sistema permite duplicar la estructura del proyecto en una nueva ruta.

4.3.1.2 Requerimientos funcionales de pasos

A continuación se detallan los requerimientos funcionales de los pasos:

Requerimientos
El sistema permite dar de alta pasos.
El sistema permite modificar los pasos introducidos en el sistema.
El sistema permite asignar y desasignar los scripts definidos en el diccionario a los pasos.

4.3.1.3 Requerimientos funcionales de scripts

Los requerimientos de los scripts son parecidos a los requerimientos de los pasos:

Requerimientos
El sistema permite dar de alta scripts.
El sistema permite modificar la información asociada al script.
El sistema permite asignar y desasignar los comandos definidos en el diccionario a los scripts.

4.3.1.4 Requerimientos funcionales de comandos

Estos son los requerimientos funcionales para los comandos:

Requerimientos
El sistema permite definir nuevos comandos.
El sistema permite modificar los comandos guardados en el sistema.
El sistema permite que los comandos utilicen variables definidas en el sistema.

4.3.1.5 Requerimientos funcionales de procedimientos del sistema

Estos requerimientos se clasifican de esta manera por no estar vinculados directamente con alguna entidad propia del sistema:

Requerimientos
El sistema permite generar una estructura de experimentación parecida a la que se dispone en el sistema actual mediante las diferentes entidades componentes en el sistema.
La estructura que genera el sistema se puede regenerar tantas veces se quiera.
El sistema permite copiar entidades de otros proyectos.

4.3.2 Requerimientos no funcionales

Los requerimientos no funcionales de un sistema, a veces, pueden resultar decisivos en un sistema. Son cualidades que debe tener el sistema para que, junto a los requerimientos funcionales, el sistema sea eficaz.

Los requerimientos no funcionales de este sistema son:

- Escalabilidad: el sistema debe ser construido sobre la base de un desarrollo evolutivo e incremental de manera tal que si aparecen o se desean añadir nueva funcionalidades y requerimientos relacionados puedan ser incorporados afectando al sistema existente de la menor manera posible. Se pretende, por tanto, incorporar aspectos de reutilización de código.

- Flexibilidad: el sistema debe ser diseñado y construido con los mayores niveles de flexibilidad en cuanto a la parametrización de los tipos de datos, de tal manera que la administración del sistema sea realizada de manera sencilla.

- Usabilidad: la interfaz del sistema será amigable e intuitiva. Se pretende que todo el sistema sea homogéneo con la finalidad de la rápida comprensión de su estructura.

- Instalación: El sistema debe ser fácil de instalar en todas las plataformas de hardware y software de posible uso (por ejemplo Linux y Unix).

- Arquitectura: La solución debe permitir el libre acceso a los datos de una manera sencilla, sin necesidad de ningún software adicional al de cualquier sistema, ni tan sólo el propio sistema a desarrollar, para acceder a los datos.

5. PLANIFICACIÓN Y COSTES

Una vez que se han definido los objetivos y los requisitos funcionales y no funcionales que tiene que satisfacer el sistema a desarrollar, la planificación definirá de enumerar las diferentes tareas necesarias para construir este sistema.

En muchos de proyectos la planificación sólo sirve como orientación a los pasos a seguir para llegar a las metas establecidas. Esta orientación podría ser cualquiera de los puntos siguientes:

- Asegurar el proceso de desarrollo. Se establece una estimativa de los recursos y el tiempo necesario para desarrollar el sistema.
- Identificar, mediante la planificación realizada al inicio, las desviaciones y retardos en las fechas de entrega, para poder analizar la causa.
- Facilitar la toma de decisiones y establecer prioridades en las tareas.
- Tener una referencia para planificar futuros proyectos y realizar cálculos de tiempos y recursos más precisos.
- Servirá como base para realizar una estimación económica del proyecto.

5.1 Planificación

La realización de una planificación en la mayoría de proyectos dónde se ven implicados la gestión de recursos, es importante de cara a que la construcción del sistema establecido por los objetivos y el análisis de requerimientos se convierta en algo eterno.

Existe en el mercado software específica para realizar la planificación, que utilizan diversas técnicas. Realizan además cálculos de costes, gestión de recursos, asignación de tareas...

No se considera oportuno en este documento incluir una planificación realizada con una de estas herramientas por considerar que en este proyecto interviene una persona para realizar las tareas que un proyecto de gran envergadura intervendrían varias, con la consecuente repartición de tareas y la definición de tiempos concurrentes.

En su lugar se identificará la planificación por etapas de las tareas a realizar y la estimación de tiempo correspondiente de cada etapa.
Para definir las etapas se aplicará un análisis descendente:

- Etapa de desarrollo
- Etapa de reestructuración

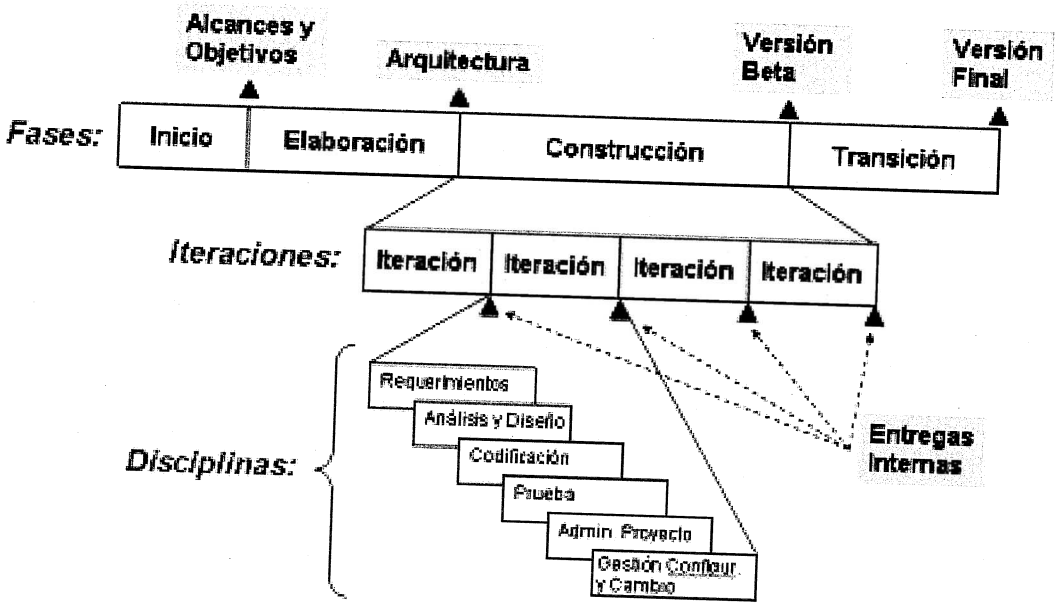
5.1.1 Desarrollo

En este apartado se describen más detalladamente las diferentes tareas a realizar para conseguir los objetivos marcados. Se incluye junto a cada tarea una estimación de tiempo.

TAREAS	**TIEMPO**
Inicio	4 días
Toma de contacto	1
Realización de las primeras entrevistas	3
Análisis de requerimientos	9 días
Sistema vigente	2
Requerimientos funcionales	4
Requerimientos no funcionales	3
Especificación	4 días
Modelo conceptual	4
Diseño	24 días
Arquitectura	4
Definición modelo conceptual arquitectura	4
Gestión de datos	4
Dominio	6
Presentación	6
Implementación	53 días
Implementación de la gestión de datos	15
Implementación del dominio	10
Implementación de la capa de presentación	25
Corrección de errores	3
TOTAL	**94 días**

5.1.2 Reestructuración

Normalmente, en todos los proyectos, con el análisis de requisitos y la planificación debería ser suficiente para poder construir el sistema. Sin embargo, en el ciclo de vida del proyecto, aparecen correcciones que pueden replantear fases del proyecto, como se puede ver en el siguiente gráfico:



Se tiene en cuenta, por tanto, alguna iteración dentro de la planificación, partiendo de la base que uno de los requerimientos no funcionales iniciales es la escalabilidad:

TAREAS	**TIEMPO**
Análisis de requerimientos iteración	1 días
Requerimientos funcionales y no funcionales	1
Especificación iteración	1 días
Modelo conceptual	1
Diseño iteración	1 días
Arquitectura, Dominio y presentación	1
Implementación iteración	6 días
Implementación de la gestión de datos y el dominio	1
Implementación de la capa de presentación	4
Corrección de errores	1
TOTAL	**9 días**

Se prevén que como mucho pueda haber 1 ó 2 iteraciones, por tanto se tendrán un margen de 9 días.

5.1.3 Estimación de duración total

Una vez descritas las dos fases (la de desarrollo y la de reestructuración), se establece el tiempo del proyecto, teniendo en cuenta que como recursos sólo se dispone de una persona. No hay concurrencia y todas las tareas se deberán hacer de una manera secuencial.

El intervalo de tiempo previsto para realizar el proyecto es de 94 días para la parte de desarrollo más los 18 días (en el peor de los casos) que suma un total de 112 días. Si se tiene en cuenta una media de 22 días laborables por mes, sale una duración de unos 5 meses.

5.2 Estudio económico

La planificación no sólo aporta información sobre el tiempo y los recursos. Una vez realizada permite hacer una estimación económica de cuanto puede costar el proyecto.

En una empresa de desarrollo de software es la herramienta para justificar a un cliente que lo que compra tiene el valor que se establece, sobretodo en proyectos de gran envergadura, donde se utilizan muchos recursos para construir el sistema.

Para realizar el estudio económico hay que basarse en todos los elementos participantes en el proyecto que nos suponen un gasto (se entiende un gasto variable). Se distinguen dos tipos de gastos:

- Los ocasionados por las diferentes personas, que actúan con rol determinado en el proyecto.
- El coste del hardware y software utilizado en el proyecto.

5.2.1 Costes de desarrollo

Se ha hablado que cada persona de las que participa puede hacerlo con uno o varios roles. Los roles dependen de la responsabilidad de cada persona y las tareas que se desempeñan. En base a esto establecemos el precio por roles desempeñados.

En el coste anual se establece un cargo adicional al sueldo bruto anual de un 33%, para reflejar impuestos que paga una empresa por un trabajador y para el coste por día se establece unos 216 días (365 días – 15 festivos – 30 vacaciones – 104 de fines de semana). Se insiste en que toda esta planificación es orientativa.

Rol	Sueldo bruto anual	Coste anual	Coste/día
Jefe de proyecto	45.000 €	59.850 €	277 €
Analista	35.000 €	46.550 €	215 €
Programador	22.000 €	29.260 €	136 €

Establecido el coste de cada rol y teniendo en cuenta la planificación realizada se detalla a continuación el coste de cada tarea. Se ha asignado al jefe de proyecto una media de dos días por mes de proyecto, al analista se le ha asignado las tareas de inicio, análisis de requisitos y especificación. Al programador el resto de tareas.

Rol	Días	Coste diario	Coste
Jefe de proyecto	10 días	277 €	2.770 €
Analista	17 días	215 €	3.655 €
Programador	77 días	136 €	10.472 €
TOTAL	104 días		16.897 €

A esta estimación no se le ha sumado el coste de las reestructuraciones, pero haciendo la estimación de una reestructuración, si se tiene que hacer nos costaría:

Rol	Días	Coste diario	Coste
Jefe de proyecto	2 días	277 €	554 €
Analista	2 días	215 €	430 €
Programador	7 días	136 €	952 €
TOTAL	11 días		1.936 €

Se han planificado dos reestructuraciones, con lo que el coste total de las reestructuraciones que se prevén es de 3.872 €.

5.2.2 Costes de hardware y software

A los costes del anterior apartado se han de añadir los costes de los recursos de hardware y software utilizados.

Para la realización del proyecto se ha utilizado un ordenador propio, aunque partiendo de la base que la facultad facilita esta herramienta se contabilizará como coste 0.

El software que se ha usado para desarrollar el sistema es el siguiente:

- Sistema operativo Ubuntu Linux 5.04
- Xstream 1.1.2, para la gestión de archivos XML
- Diferentes versiones de Java SE. La última JDK 1.5.0_9
- Netbeans 5.0
- OpenOffice

No se tiene en cuenta otro tipo de software como el utilizado para desarrollar este documento.

El coste de la lista de software especificado anteriormente es 0, ya que las licencias de todos ellos así lo especifican.

Una vez hecha la estimación de todos los costes se establece, que el coste real del sistema viene determinado por el coste de desarrollo, o sea, los 16.897 € más los 3.872 € de la previsión de reestructuración. El total es 20.769 €.