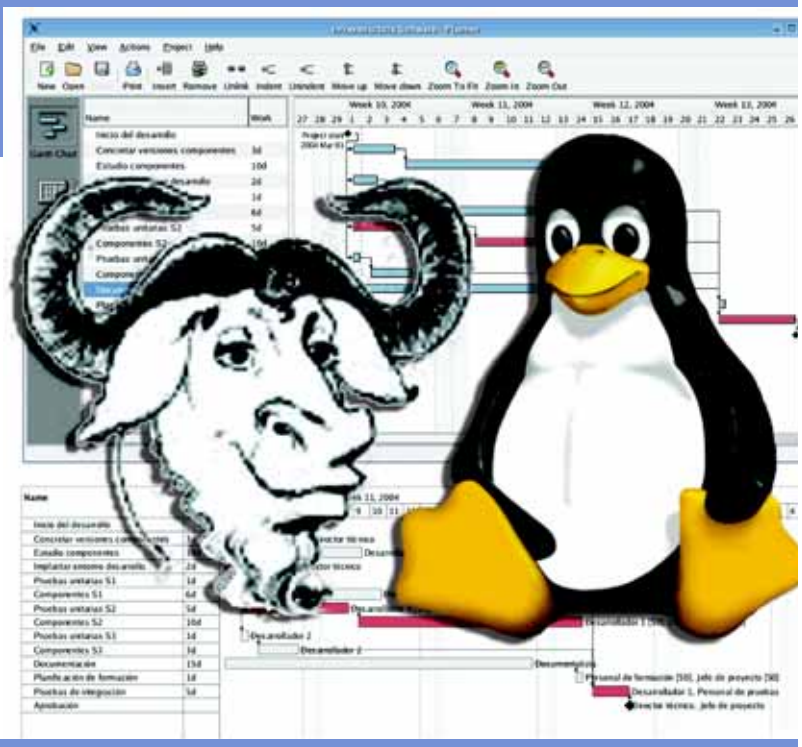
Software libre



Alberto Otero García

XP06/M2121/02158

Proyecto de dirección de sistemas de información

UOC
www.uoc.edu

David Megías Jiménez

Coordinador

Ingeniero en Informática por la UAB.

Magíster en Técnicas Avanzadas de Automatización de Procesos por la UAB.

Doctor en Informática por la UAB.

Profesor de los Estudios de Informática y Multimedia de la UOC.

Jordi Mas

Coordinador

Ingeniero de software en la empresa de código abierto Ximian, donde trabaja en la implementación del proyecto libre Mono. Como voluntario, colabora en el desarrollo del procesador de textos Abiword y en la ingeniería de las versiones en catalán del proyecto Mozilla y Gnome. Es también coordinador general de Softcatalà. Como consultor ha trabajado para empresas como Menta, Telépolis, Vodafone, Lotus, eresMas, Amena y Terra España.

Alberto Otero García

Autor

Ingeniero en Informática por la Universidad Ramon Llull. Profesor titular de la asignatura Administración de Sistemas Operativos en Enginyeria i Arquitectura La Salle. Socio fundador y jefe de proyectos de Cometa Technologies, empresa dedicada a dar soluciones en tecnologías de la información, basadas en el uso de estándares y herramientas de código abierto.

Segunda edición: febrero 2007

Av. Tibidabo, 39-43, 08035 Barcelona
Material realizado por Eureca Media, SL

Índice

Agradecimientos

El autor agradece a la Fundación para la Universitat Oberta de Catalunya (http://www.uoc.edu) la financiación de la primera edición de esta obra, enmarcada en el Máster Internacional en Software Libre ofrecido por la citada institución.

ANOTACIONES

Introducción

Para llevar a cabo un proyecto de sistemas de información en entornos de software libre, como en cualquier otro tipo de proyecto, es necesario seguir un proceso que nos lleve desde la comprensión del alcance del problema que queremos solventar hasta la implantación y mantenimiento de la solución que hayamos elegido.

Aunque el director de sistemas de información en entornos de software libre, o de un proyecto concreto basado en la utilización de software libre, no necesita conocer todas y cada una de las técnicas y herramientas utilizadas a lo largo de los proyectos, sí que necesita conocer qué fases se deberán seguir, qué productos se deben obtener al final de cada una de ellas y, a rasgos generales, cómo obtenerlos, con el fin de poder:

- Planificar. El director de proyecto debe planificar qué recursos hay que asignar a cada una de las fases del proyecto, estimar el tiempo que llevará completarlas, su coste económico, etc.

- Organizar. El director de proyecto debe poder organizar los recursos de los que dispone de la forma más óptima, coordinar el avance de dicho proyecto con el resto de proyectos de sistemas de la información y de la empresa, alinear los objetivos del proyecto con los de la organización, conocer qué herramientas son las más indicadas para su uso, etc.

- Controlar. El director de proyecto debe poder testar la buena marcha del proyecto, comprobar la calidad de los resultados obtenidos, ofrecer ayuda a los integrantes del equipo en cualquiera de las fases en caso de ser necesario, etc.

En este curso se pretenden repasar aquellas fases que es necesario seguir a lo largo de todo proyecto de sistemas de la información, y que el director de proyecto deberá supervisar.

ANOTACIONES

Estas fases son las siguientes:

- Estudio de viabilidad: se estudiará en líneas generales qué problemas se desean resolver, qué soluciones posibles existen y cuál de ellas es la más adecuada.

- Análisis: se describirá detalladamente el sistema que se desea construir, qué requisitos debe cumplir y a qué usuarios debe satisfacer.

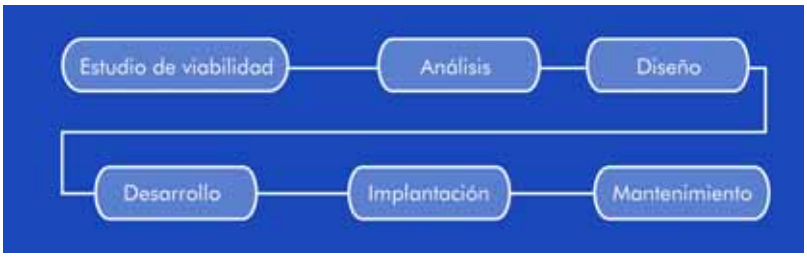
- Diseño: se realizará el planteamiento tecnológico de la solución.

- Desarrollo: se llevará a cabo la programación, integración, instalación, etc. de los diferentes subsistemas que compongan el proyecto.

- Implantación: se pasará el sistema construido a producción con el fin de que los usuarios de éste empiecen a utilizarlo.

- Mantenimiento: se realizarán tanto las correcciones de los posibles errores que puedan surgir en el sistema implantado, como las mejoras evolutivas que se consideren oportunas.

Figura 1. Fases de un proyecto de sistemas de información



Estas fases estarán presentes, de una u otra forma, con estos nombres o con otros, en cualquier proyecto de sistemas de información, desde los gestionados mediante el método "clásico" (por ejemplo, en fases seguidas secuencialmente, en cascada, etc.), hasta los gestionados como sugiere el conjunto de metodologías conocidas como ágiles.

Nota

A lo largo de todo el material del curso, se desarrolla un caso práctico con el fin de ejemplificar las explicaciones dadas. Este caso práctico no constituye de ningún modo un estudio exhaustivo del proyecto propuesto, sino que simplemente sirve como marco para ofrecer diferentes ejemplos de las partes de que se compone con fines únicamente didácticos.

ANOTACIONES

Objetivos

Los objetivos que el lector deberá alcanzar al finalizar el curso de Proyecto de dirección de sistemas de información son los siguientes:

- Haber comprendido de manera global lo que representa la dirección de un proyecto de sistemas de información, en especial en un entorno tecnológico de software libre.

- Haber asimilado qué fases integran un proyecto de sistemas de información, y qué tareas se deben llevar a cabo en cada una de ellas, especialmente desde el punto de vista de su dirección.

- Haber reflexionado sobre qué herramientas de software libre pueden ayudar en cada una de las fases de un proyecto de sistemas de información.

- Haber aplicado a un caso práctico los conocimientos adquiridos a lo largo de todo el máster en dirección de sistemas de información.

ANOTACIONES

1. Estudio de viabilidad

El objetivo de la realización del **estudio de viabilidad** es el de, dado un conjunto de necesidades planteadas, elegir aquella solución que mejor las cubra de entre todas las posibles (o descartarlas todas en caso de que ninguna las satisfaga).

En el estudio de viabilidad se considerarán las diferentes soluciones posibles, teniendo en cuenta:

- El estado inicial del sistema.
- La situación actual.
- Los requisitos planteados.

Cada una de las soluciones propuestas en el estudio de viabilidad deberá recoger los siguientes aspectos:

- Económicos: se deberá incluir un estudio económico preliminar que contemple los costes asociados a cada una de las soluciones.
- Técnicos: se deberá incluir un estudio técnico preliminar de cada una de las soluciones.
- Legales: se deberá incluir un estudio de aquellos aspectos legales que puedan influir en la viabilidad de la solución.
- Operativos: se deberá incluir un estudio previo de la operativa de cada una de las soluciones propuestas.

Una vez planteadas cada una de las soluciones, se elegirá la mejor teniendo en cuenta:

- El impacto en la organización.
- La inversión que hay que realizar.
- Los riesgos asociados.

Los siguientes apartados describen con más detalle cada una de las tareas a llevar a cabo para realizar el estudio de viabilidad.

ANOTACIONES

1.1. Establecimiento del alcance del sistema

En esta fase del estudio de viabilidad se pretende **estudiar el alcance de las necesidades planteadas por el cliente** (bien sean terceros, en caso de tratarse de un proyecto dirigido a otras organizaciones, bien sean usuarios internos, en caso de tratarse de un proyecto para la propia organización).

Lo primero que será necesario hacer es la **descripción general** de las necesidades planteadas por el cliente. En esta descripción general se deberán incluir los aspectos básicos descritos en el anterior apartado (económicos, técnicos, legales y operativos) que tengan especial relevancia.

Caso práctico

Renovación de la infraestructura software de Soluciones Abiertas, S.A.

La empresa Soluciones Abiertas, S.A., dedicada al desarrollo de software, ha decidido renovar todo el hardware del que disponen:

- 4 servidores dedicados a dar soporte a toda la empresa.
- 20 ordenadores personales, en los que se distinguen dos perfiles de uso: el del personal no técnico (ofimática, navegación por Internet, etc.) y el del personal técnico (programación, test de software, navegación por Internet, etc.).
- 1 sistema de copias de seguridad en cinta.

Junto con el cambio de hardware, se está considerando realizar también una renovación o ampliación del software usado en la infraestructura básica de la empresa, con el fin de actualizarlo y decrementar los costes de mantenimiento en la medida de lo posible. Dicho software es el siguiente:

- Sistema de almacenamiento y compartición de ficheros: es el repositorio central de archivos, donde cada usuario (técnico o no) guarda los ficheros de los cuales quiere que exista copia de seguridad y que permite compartir documentos en directorios ordenados por grupos de trabajo.

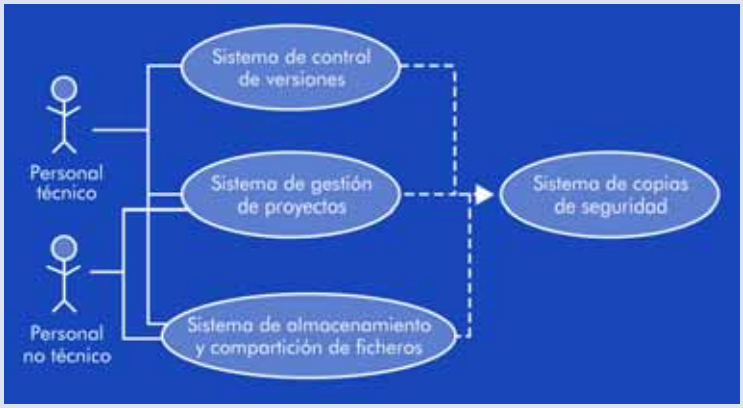
ANOTACIONES

- Sistema de control de versiones: en la actualidad no existe, y sería utilizado por el personal técnico con el fin de mantener las versiones, cambios entre éstas, etc. del código fuente de los diferentes productos software de la empresa.

- Sistema de copias de seguridad: es el encargado de realizar los *backups* totales e incrementales de la información que se considera más valiosa en la empresa (la base de datos del sistema de gestión de proyectos, los ficheros de usuarios técnicos y no técnicos, y el código almacenado en el sistema de control de versiones).

Desde el punto de vista económico, para que la renovación sea viable deberá implicar el menor gasto posible, dado que la partida presupuestaria más importante correspondiente a sistemas de información se va a destinar a la renovación del hardware. A nivel técnico, las necesidades planteadas son muy poco restrictivas, ya que al ser Soluciones Abiertas una empresa dedicada a las tecnologías de la información, se dispone de personal cualificado que afronta y disfruta cualquier reto técnico sin mayores problemas. Desde el punto de vista legal, se exige que las soluciones aportadas sean lo más flexibles posibles, ya que se valora muy negativamente el hecho de no disponer de la máxima libertad para copiar y/o modificar los sistemas software que se implanten. A nivel operativo, la única necesidad planteada consiste en que en ningún caso se debe perder funcionalidad de la que en estos momentos ya se dispone, sino en todo caso, ampliarla.

Figura 2. Descripción general del sistema



ANOTACIONES

Asimismo, además de describir de forma general el proyecto, se debe tener en cuenta **cómo afectará** éste a:

- Otros proyectos de tecnologías de la información ya en curso o que se piensan poner en marcha.
- Las diferentes unidades de la organización, teniendo en cuenta quiénes son los responsables de éstas y cuál es su estructura.

Caso práctico

Alcance del proyecto de renovación de la infraestructura software de Soluciones Abiertas, S.A.

En el caso de Soluciones Abiertas, S.A., el proyecto de renovación de la infraestructura software afectará a:

- El proyecto de renovación del hardware de la empresa. Según el software de base escogido, será más apropiado un hardware u otro, haciéndose necesaria la coordinación de ambos proyectos.
- El proyecto de renovación de los entornos integrados de programación (IDEs) utilizados por el personal técnico, ya que de manera ideal éste se debería integrar totalmente con el sistema de control de versiones.

Por otro lado, el cambio de la infraestructura software de la empresa afectará a los siguientes departamentos de ésta:

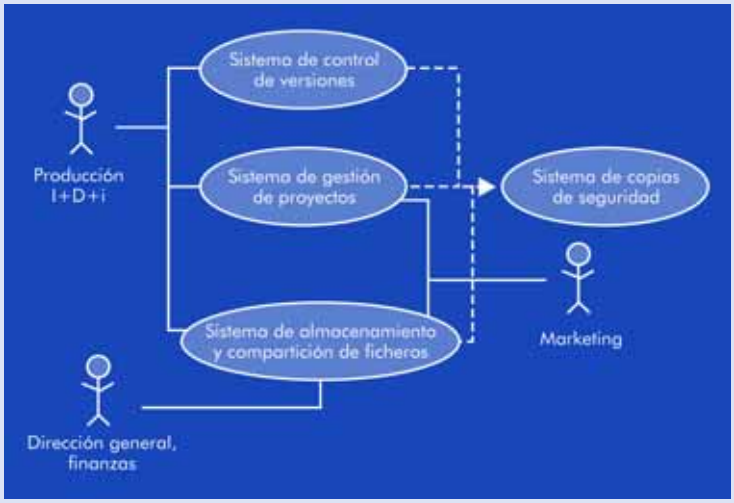
- Dirección general, finanzas: estos departamentos se verán afectados únicamente durante el proceso de migración a las nuevas plataformas, ya que en algún momento la disponibilidad del sistema de almacenamiento de ficheros puede verse afectada. Por lo demás, la influencia de la renovación debería ser mínima.
- Marketing: este departamento se verá afectado, al igual que dirección general y finanzas, por la renovación del sistema de almacenamiento de ficheros. Además, se verán afectados por el cambio del sistema de gestión de proyectos, debido a que las personas de esta unidad de la empresa son las que gestionan el contacto con el cliente y por tanto son usuarios habituales del mismo (ya que

ANOTACIONES

les permite introducir peticiones de mejora de los productos, descripciones de *bugs* en éste, etc.).

- Producción, investigación y desarrollo (I+D): estos dos departamentos serán los más afectados por el cambio, ya que para ellos implica, además de lo que se ha comentado para los anteriores departamentos, la incorporación de la utilización del sistema de control de versiones en el proceso productivo y de investigación.
- Se deberá informar puntualmente a los responsables de cada uno de los departamentos de los detalles y el alcance del proyecto, en la medida en que se vean afectados.

Figura 3. Descripción general del sistema según departamentos de la empresa



1.2. Estudio de la situación actual

En esta fase del estudio de viabilidad se pretende **estudiar la situación en la que se encuentra el sistema de información de la empresa y realizar un diagnóstico de éste,** siempre en lo referente al proyecto que nos ocupa.

La primera tarea que hay que realizar dentro del estudio de la situación actual es la de **identificar aquellos sistemas que deben describirse**, esto es, de qué sistemas vamos a hacer el estudio debido a que se ven

ANOTACIONES

afectados de alguna manera por el proyecto contemplado en el estudio de viabilidad. Es interesante también fijar qué usuarios participarán en el estudio de la situación actual de cada uno de los sistemas escogidos.

Caso práctico

Identificación de los sistemas actuales

Dado que el proyecto consiste en la renovación y ampliación de la infraestructura software de Soluciones Abiertas, S.A., deberemos estudiar como mínimo la situación actual de los sistemas que deseamos renovar. En este caso, como ya se ha comentado, estos sistemas serán el de gestión de proyectos, control de versiones, almacenamiento de ficheros y copia de seguridad. Con el fin de realizar un diagnóstico lo más completo posible, hemos decidido trabajar junto con usuarios de los departamentos de marketing y producción (ya que son los usuarios más intensivos de dichos sistemas). El sistema de control de versiones se obviará en esta fase del estudio de viabilidad, ya que en la actualidad la empresa carece de él.

El siguiente paso dentro del estudio de la situación actual será el de **describir cada uno de los sistemas identificados** en el paso anterior. La descripción se hará teniendo en cuenta la información recogida en las sesiones de trabajo con los usuarios seleccionados como representativos, y deberá alcanzar el nivel de detalle suficiente como para poder realizar un diagnóstico acertado del estado real de cada uno de los sistemas estudiados. Asimismo, la dedicación de recursos en esta fase dependerá de la información de partida de la cual se disponga (la descripción de la situación actual puede ser trivial en algunos casos o tremendamente complicada en otros).

Caso práctico

Descripción de los sistemas actuales

A continuación se describe la situación actual de los sistemas de Soluciones Abiertas, S.A. seleccionados con anterioridad:

- Sistema de gestión de proyectos: en la actualidad este sistema consiste en una intranet desarrollada por la propia empresa. Esta intranet está alojada en un servidor con sistema operativo GNU/Linux, servidor web Apache y progra-

ANOTACIONES

mada en PHP, apoyándose en el sistema gestor de base de datos relacional (SGBDR) MySQL.

- Sistema de almacenamiento y compartición de ficheros: en la actualidad este sistema consiste en un servidor con directorios compartidos, a los cuales puede acceder cualquier persona de la empresa. Dicho servidor tiene como sistema operativo Microsoft Windows 2000, aprovechando la funcionalidad de compartición de carpetas que éste ofrece.

- Sistema de copias de seguridad: en la actualidad este sistema consiste en el volcado diario a cinta de todos los ficheros almacenados en el sistema de almacenamiento. El servidor al cual está conectada la unidad lectora de cintas tiene como sistema operativo Microsoft Windows NT, y las copias se realizan mediante el software suministrado con el propio sistema operativo.

Figura 4. Descripción general del sistema actual

Para completar el estudio de la situación actual de los sistemas, se deberá realizar un **diagnóstico** de éstos, es decir, analizar la información obtenida detectando posibles problemas y puntos de mejora.

ANOTACIONES

Caso práctico

Diagnóstico de los sistemas actuales

Una vez analizada la información obtenida en la descripción de la situación actual de los sistemas estudiados en la empresa Soluciones Abiertas, S.A., se ha llegado a las siguientes conclusiones:

- Sistema de gestión de proyectos: se ha detectado que la funcionalidad disponible es muy limitada, ya que cada nueva característica que se desea añadir implica realizar una programación a medida (recordamos que este sistema ha sido desarrollado por la propia empresa). Este hecho hará que en el futuro o bien la herramienta quede obsoleta o bien los costes de mantenimiento de ésta sean demasiado elevados.

- Sistema de almacenamiento y compartición de ficheros: se hace patente la necesidad de crear un sistema de usuarios y permisos que permita consultar ciertos ficheros únicamente a aquellas personas o grupos de personas que estén autorizados para ello. Además, existe cierta preocupación por la escalabilidad de todo el sistema, ya que últimamente se ha notado una cierta ralentización de éste debido a que cada vez es más usado (se guardan más ficheros, se accede más a él, soporta más usuarios, etc.).

- Sistema de copias de seguridad: se ha llegado a la conclusión de que es necesario renovar la estrategia de *backup*, ya que en este momento se hacen siempre copias completas de todos los ficheros. También se ha detectado la necesidad de empezar a hacer copias de datos que hasta ahora no se hacían y que residen en servidores que no comparten directorios por ningún método de los habituales (p. ej., el servidor que alberga el sistema de gestión de proyectos).

ANOTACIONES

1.3. Definición de requisitos del sistema

Una vez descrita la situación actual del sistema, y teniendo en cuenta las opiniones de los diferentes usuarios implicados, se pasará a **describir de forma general los requisitos que deberá cumplir el proyecto** del cual se está estudiando la viabilidad.

La descripción del conjunto de requisitos a cumplir por el proyecto servirá posteriormente para evaluar cada una de las posibles soluciones alternativas existentes. Es por ello por lo que además de la citada descripción, es interesante incluir una calificación de la prioridad de cada uno de los requisitos, con el fin de tener presente su importancia relativa respecto al resto.

La descripción de cada requisito incluirá una explicación de éste, la prioridad que se le asigna y una catalogación dentro de un conjunto de categorías definidas.

Caso práctico

Definición general de requisitos del sistema de copias de seguridad de Soluciones Abiertas, S.A.

Mediante el estudio del sistema de copias de seguridad actual, los puntos de mejora y problemas detectados, y las entrevistas con los usuarios, se han identificado y catalogado los siguientes requisitos (la prioridad de cada uno de ellos está indicada como un número entre 0 y 100, siendo 100 el prioritario):

Requisitos técnicos

- (100) Arquitectura: deberán poder hacerse copias de servidores remotos conectados mediante la red al servidor de copias de seguridad.
- (100) Arquitectura: las copias de seguridad deberán poder hacerse de servidores con sistemas operativos iguales o diferentes al del servidor de copias de seguridad.
- (80) Seguridad: las copias de seguridad deben residir en un servidor que sea seguro, dado que contendrán información muy relevante para la empresa.

Nota

En el resto de este apartado nos centraremos en el estudio del sistema de copias de seguridad como ejemplo particular del caso práctico planteado hasta el momento. Para el resto de sistemas (gestión de proyectos, almacenamiento y compartición de ficheros, control de versiones), sería necesario seguir el mismo proceso que se describirá de este punto en adelante con el sistema de copias de seguridad.

ANOTACIONES

Caso práctico

(50) Normativas y/o estándares: las copias de seguridad deberán almacenarse en un formato lo más abierto posible. En la medida de lo posible se intentará que dicho formato sea independiente de la plataforma sobre la cual se hagan las copias.

Requisitos operativos

- (100) Operativa: deben poder realizarse copias de seguridad completas e incrementales.
- (80) Operativa: deben poder restaurarse ficheros concretos que residan dentro de cierta copia de seguridad, no ésta entera.
- (20) Configuración: la interfaz de programación de copias debe ser lo más intuitiva y fácil de usar posible.
- (50) Configuración: la periodicidad de realización de las copias de seguridad debe ser totalmente configurable.
- (100) Soportes: el software de copias de seguridad debe ser compatible con el hardware escogido como soporte de almacenamiento (la unidad de cintas escogida).
- (60) Soportes: el software de copias de seguridad debe ser compatible con el máximo número de soportes de almacenamiento posibles.

Requisitos legales

- (60) La licencia de uso del software de copias de seguridad debe ser lo menos restrictiva posible.
- (60) La licencia de uso del sistema operativo del servidor de copias de seguridad debe ser lo menos restrictiva posible.

Requisitos económicos

- (80) En el caso de ser necesario un gasto en concepto de licencia de uso del software de copias de seguridad, éste debe ser lo más pequeño posible.
- (80) El gasto correspondiente al sistema operativo del servidor de copias de seguridad debe ser lo más pequeño posible.

ANOTACIONES

1.4. Estudio de alternativas de solución

Una vez expresados los requisitos que deberá cumplir el proyecto del cual se está realizando el estudio de viabilidad, se pasará a **proponer varias soluciones alternativas** que cumplan con éstos. En esta fase se tendrá en consideración, asimismo, toda la información recogida hasta el momento: descripción general, alcance, situación actual, etc.

Para cada alternativa se deberá especificar en qué consiste, tanto a nivel funcional como técnico (estudiando en qué medida se cubren los requisitos descritos previamente), si está basada o no en algún producto ya existente en el mercado (en tal caso se estudiará éste, describiendo posibles costes de licencias, evolución prevista, estándares que cumple, etc.), y si supone o no la necesidad de realizar algún desarrollo a medida (en tal caso se deberá describir éste de tal modo que quede claro cuál es su alcance).

Caso práctico

Soluciones alternativas para el sistema de copias de seguridad de Soluciones Abiertas, S.A.

Como ejemplo de alternativas al sistema de copias de seguridad se proponen tres soluciones posibles, las cuales tienen las siguientes características:

- Microsoft Windows + aplicación propietaria: en este caso el sistema operativo del servidor de *backups* será Microsoft Windows 2000, y éstas se harán mediante la adquisición de un software específico de realización de copias de seguridad, por ejemplo Arkeia (http://www.arkeia.com/). Se ha comprobado que el software de copias de seguridad elegido cumple con los requisitos funcionales y técnicos definidos al respecto. En lo referente a los requisitos legales y económicos de la solución propuesta, ni el sistema operativo ni el software de copias de seguridad cumplen con lo expresado.

ANOTACIONES

- GNU/Linux + aplicación propietaria: en este caso el sistema operativo del servidor de *backups* sería GNU/Linux (en cualquiera de sus distribuciones), sin embargo, el software de copia de seguridad sería propietario, por ejemplo Arkeia. Se ha comprobado que el software de copias de seguridad elegido cumple con los requisitos funcionales y técnicos definidos al respecto. En lo referente a los requisitos legales y económicos, el sistema operativo cumple con ellos (ya que no es necesario pagar ninguna licencia de uso, y además se nos permite hasta estudiar y modificar el código fuente de éste), pero no así el software de copias de seguridad (ya que éste es propietario).

- GNU/Linux + aplicación libre: en este caso, tanto el sistema operativo del servidor de *backups* como el software de copias de seguridad serían libres (por ejemplo, Amanda, http://www.amanda.org/). Se ha comprobado que el software de copias de seguridad elegido cumple con los requisitos funcionales y técnicos definidos al respecto, salvo el de ofrecer una interfaz de fácil utilización. En cuanto a los requisitos legales y económicos, éstos se cubren perfectamente, ya que las licencias son muy flexibles y los costes de adquisición son nulos.

En los tres casos será necesaria la realización de un módulo de software que recoja la información almacenada en la base de datos del sistema de gestión de proyectos y la almacene en ficheros que puedan ser copiados al sistema de *backups*.

Sobre la base de la información obtenida por diferentes canales (informes, foros de discusión, experiencia del personal de la empresa, etc.) se ha considerado que el coste de instalación y mantenimiento de los componentes es igual en los tres casos.

ANOTACIONES

1.5. Valoración de las alternativas

Una vez se han estudiado las soluciones alternativas dentro del proyecto del cual se está haciendo el estudio de viabilidad, se debe pa-

sar a **valorarlas considerando su viabilidad económica (análisis costes/beneficios) y riesgos que comportan**.

Para cada una de las posibles soluciones, se deberá **estudiar su viabilidad económica**, esto es, confeccionar un análisis costes/beneficios que deje patente el gasto que será necesario realizar y lo que se espera obtener a cambio (tanto de forma tangible como intangible).

Caso práctico

Análisis costes/beneficios del sistema de copias de seguridad de Soluciones Abiertas, S.A.

Los costes de adquisición imputados a cada una de las soluciones son:

- Microsoft Windows + aplicación propietaria = = 550 € + 500 € = 1.050 €
- GNU/Linux + aplicación propietaria = 0 € + 500 € = 500 €
- GNU/Linux + aplicación libre = 0 € + 0 € = 0 €

Dado que consideramos que los costes de instalación y mantenimiento son los mismos para los tres casos, éstos no tendrán efecto en la comparación que estamos realizando (en un caso real podrían tener mucha importancia, ya que no únicamente se compararían las diferentes soluciones, sino que se estudiaría la viabilidad económica de elegir cualquiera de ellas).

En el caso de la tercera opción, GNU/Linux + aplicación libre, debemos tener en cuenta el coste añadido asociado al aprendizaje en la utilización de la aplicación libre de copias de seguridad, dado que suponemos que la interfaz de usuario no es tan fácil de manejar como la de la aplicación propietaria. Suponemos que el hecho de utilizar la aplicación libre frente a la aplicación propietaria requerirá 10 horas extra de dedicación (a un precio medio de 30 €/hora) acerca de la utilización de la primera. Es necesario sumar por tanto este coste al de adquisición: 0 € + 300 € = 300 €.

Los beneficios de cada una de las soluciones son los descritos en apartados anteriores (requisitos, descripción, etc.).

Nota

Los precios indicados son ficticios, utilizados únicamente a modo de ejemplo.

ANOTACIONES

Además de estudiar la viabilidad económica de las diferentes soluciones, deberemos tener en cuenta los **riesgos asociados** a cada una de ellas. Para cada una de las alternativas existentes describiremos qué incertidumbres, problemas potenciales, etc. existen.

Caso práctico

Riesgos en las alternativas del sistema de copias de seguridad de Soluciones Abiertas, S.A.

Los riesgos asociados a cada una de las soluciones alternativas son los siguientes:

Microsoft Windows + aplicación propietaria:

- Sistema operativo: cambio en la estrategia de negocio del fabricante, desapareciendo el soporte dado hasta el momento y haciéndose necesaria una actualización.
- Sistema operativo: fallos de seguridad detectados pero no subsanados por el fabricante en un periodo de tiempo razonable.
- Aplicación propietaria: desaparición del fabricante del producto, o cambio de estrategia de negocio (dado que no se dispone del código fuente de la aplicación, este hecho supondría que cualquier error o problema no podría ser subsanado).

GNU/Linux + aplicación propietaria:

- Sistema operativo: se podría dar la falta de soporte en determinados casos, ya que no existe un sólo fabricante que centralice el desarrollo del sistema operativo.
- Aplicación propietaria: desaparición del fabricante del producto, o cambio de estrategia de negocio (dado que no se dispone del código fuente de la aplicación, este hecho supondría que cualquier error o problema no podría ser subsanado).

ANOTACIONES

GNU/Linux + aplicación libre:

- Sistema operativo: se podría dar la falta de soporte en determinados casos, ya que no existe un solo fabricante que centralice el desarrollo del sistema operativo.

- Aplicación propietaria: desaparición del equipo principal de desarrolladores que mantienen la aplicación.

Llegados a este punto, se deberá realizar una **propuesta de enfoque con el fin de paliar en la medida de lo posible los riesgos** antes descritos. De este modo intentaremos reflejar si estos riesgos son salvables, haciéndose relevante su importancia relativa.

Caso práctico

Paliación de riesgos en las alternativas del sistema de copias de seguridad de Soluciones Abiertas, S.A.

Los posibles enfoques con el fin de paliar los riesgos asociados a cada una de las soluciones alternativas son los siguientes:

Microsoft Windows + aplicación propietaria:

- Sistema operativo: firma de contrato de soporte del sistema operativo con el fabricante de éste por un periodo de tiempo igual al que estimemos que será la vida del sistema de seguridad tal y como lo estamos estudiando. Esta solución debe ser aceptada por el fabricante para poder llevarse a cabo.

- Sistema operativo: firma de contrato de soporte con indemnizaciones en caso de producirse fallos en la seguridad del sistema debido a problemas en el sistema operativo. Esta solución debe ser aceptada por el fabricante para poder llevarse a cabo.

- Aplicación propietaria: firma de contrato en el cual el fabricante se comprometa a suministrar al menos el código fuente de su aplicación en caso de que cese su actividad.

ANOTACIONES

GNU/Linux + aplicación propietaria:

- Sistema operativo: puede contratarse el soporte de una empresa externa que se comprometa a centralizar y resolver los posibles problemas que puedan surgir.
- Aplicación propietaria: firma de contrato en el cual el fabricante se comprometa a suministrar al menos el código fuente de su aplicación en caso de que cese su actividad.

GNU/Linux + aplicación libre:

- Sistema operativo: puede contratarse el soporte de una empresa externa que se comprometa a centralizar y resolver los posibles problemas que puedan surgir.
- Aplicación propietaria: se debe valorar la estabilidad y alcance de la comunidad formada en torno a la aplicación, ya que en caso de que el equipo principal de desarrolladores desaparezca, la continuidad de ésta dependerá del número de personas que la utilizan y desarrollan esporádicamente en todo el mundo.

1.6. Selección de la solución

Para acabar con el estudio de viabilidad, se **elegirá una solución de entre las diferentes alternativas estudiadas**.

La decisión sobre cuál es la mejor solución (o si ninguna lo es) se tomará teniendo en cuenta la información acumulada hasta el momento:

- Descripción general y alcance del proyecto.
- Situación actual del sistema.
- Requisitos que deberá cumplir la solución adoptada.
- Descripción de las soluciones alternativas consideradas.
- Análisis de costes/beneficios de las diferentes soluciones y riesgos asociados a cada una de ellas.

ANOTACIONES

Caso práctico

Selección de la solución adoptada en el sistema de copias de seguridad de Soluciones Abiertas, S.A.

Dada la descripción general del sistema y la situación actual de éste, se han considerado los siguientes factores con el fin de realizar la elección de la solución:

- Requisitos planteados y descripción de cada una de las soluciones: todas las soluciones cubren en mayor o menor medida los requisitos básicos a nivel funcional y técnico. En cuanto a los aspectos económicos y legales, la solución GNU/Linux + aplicación libre es la clara ganadora.

- Análisis costes/beneficios: este análisis ha dado como resultado tres costes entre los cuales la solución GNU/Linux + aplicación libre es la más barata. Dado que los beneficios aportados por cada solución son parecidos en términos generales (desde luego podrían discutirse ciertos detalles en los que sí hay diferencias significativas, pero que no decantan definitivamente la balanza por una u otra solución), se ha optado por valorar como más positiva la solución GNU/Linux + aplicación libre.

- Riesgos: se han detectado posibles problemas de diferentes tipos en cada una de las soluciones, siendo los de más fácil solución los relacionados con el sistema operativo GNU/Linux y la aplicación de copias de seguridad libre (precisamente por su carácter marcadamente abierto en comparación con el resto).

Se decide por tanto que la solución en GNU/Linux + aplicación libre de copias de seguridad es la más adecuada de entre todas las consideradas.

ANOTACIONES

2. Análisis del sistema

El objetivo de la realización del **análisis del sistema** es el de, dada la solución escogida de entre las descritas en el estudio de viabilidad, llevar a cabo una especificación detallada de ésta (orientada a facilitar el diseño del sistema, fase cubierta en el siguiente capítulo).

Los siguientes apartados describen con más detalle cada una de las tareas que cabe efectuar para realizar el análisis del sistema.

2.1. Definición del sistema

En esta fase del análisis se deberá **describir el sistema**, establecer **cómo se comunicará con otros** en caso de ser necesario y **qué usuarios serán representativos** en el uso del mismo.

Como el lector recordará, ya en la fase de estudio de viabilidad se procedió a describir el sistema genéricamente, así como a definir cómo afectaba éste al resto de sistemas ya existentes (o proyectos que se pensaba llevar a cabo). El trabajo realizado en dicha fase servirá como base de las tareas realizadas en el presente análisis.

Utilizando como punto de partida la descripción de los requisitos hecha en el estudio de viabilidad, se **determinarán los requisitos exactos del sistema**. Asimismo, se estudiará **cómo se comunica el sistema** con el resto de sistemas existentes (ya sea recibiendo o enviado información a éstos).

ANOTACIONES

Caso práctico

Requisitos exactos del sistema de copias de seguridad de Soluciones Abiertas, S.A.

El sistema de copias de seguridad de Soluciones Abiertas, S.A. deberá cumplir los siguientes requisitos:

- Deberán poder hacerse copias de servidores remotos conectados mediante la red local de la empresa al servidor de copias de seguridad.
- Las copias de seguridad deberán poder hacerse de servidores con sistemas operativos iguales (GNU/Linux) o diferentes al del servidor de copias de seguridad (otros UNIX o Microsoft Windows).
- Las copias de seguridad deben residir en un servidor que sea seguro, dado que contendrán información muy relevante para la empresa. Este servidor deberá cumplir las normas de seguridad fijadas para todos los servidores de la empresa.
- Las copias de seguridad deberán almacenarse en un formato lo más abierto posible. En la medida de lo posible, se intentará que dicho formato sea independiente de la plataforma sobre la cual se hagan las copias.
- Deben poder realizarse copias de seguridad completas (p. ej., una a la semana) e incrementales (p. ej., una al día).
- La interfaz de programación de copias debe ser lo más intuitiva y fácil de usar posible. Esta interfaz podría consistir en una serie de ficheros de texto modificables por los administradores de las copias de seguridad.
- Deben poder restaurarse ficheros concretos que residan dentro de cierta copia de seguridad, no ésta entera. Este proceso se llevará a cabo mediante la interfaz del software

ANOTACIONES

de copias de seguridad, eligiendo aquellos ficheros que deseemos restaurar.

- La periodicidad de realización de las copias de seguridad debe ser totalmente configurable mediante la interfaz de programación de copias de seguridad.

- El software de copias de seguridad debe soportar el hardware elegido como soporte de almacenamiento (la unidad de cintas elegida será la comprada en el proyecto de renovación de hardware).

- La licencia de uso del software de copias de seguridad debe ser lo menos restrictiva posible, en concreto deberá ser de código abierto o libre.

- La licencia de uso del sistema operativo del servidor de copias de seguridad será la correspondiente a GNU/Linux, es decir, GNU General Public License.

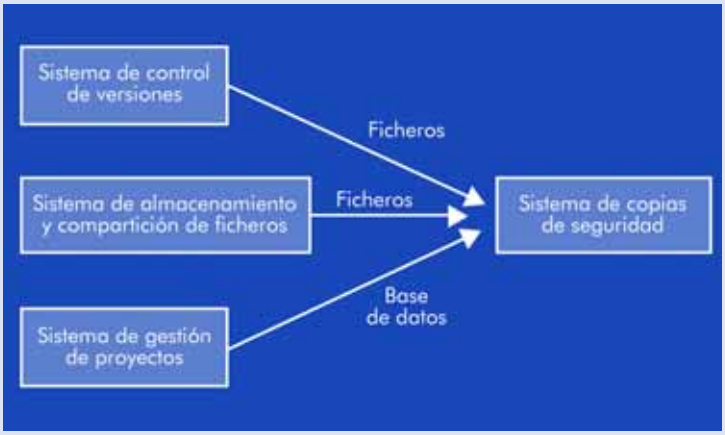
- El gasto en concepto de licencia de uso del software de copias de seguridad debe ser nulo.

- El gasto correspondiente al sistema operativo del servidor de copias de seguridad debe ser nulo.

- Las comunicaciones existentes con otros sistemas (almacenamiento y compartición de ficheros, gestión de proyectos y control de versiones) consistirán en la recogida de la información de la cual se debe hacer *backup* por parte del servidor de copias de seguridad. En el primer y tercer caso (sistema de almacenamiento y compartición de ficheros, control de versiones), será necesario indicar qué archivos se desean copiar y con qué periodicidad. En el segundo caso (gestión de proyectos), el objetivo básico será realizar una copia de seguridad de la base de datos, con lo que será necesario guardar ésta como un archivo que pueda copiarse como el resto.

ANOTACIONES

Figura 5. Descripción general del sistema de copias de seguridad



En la definición del sistema también será necesario definir el **entorno tecnológico** del proyecto, respecto del cual ya se incluyó información en el estudio de viabilidad.

Caso práctico

Entorno tecnológico del sistema de copias de seguridad de Soluciones Abiertas, S.A.

El entorno tecnológico del sistema de copias de seguridad será el siguiente:

- Sistema operativo: GNU/Linux (distribución por determinar).
- Sistema de copias de seguridad: deberá poder ejecutarse en el sistema operativo GNU/Linux y estar hecha en un lenguaje que el equipo de personas de Soluciones Abiertas, S.A. conozca (p. ej., C, C++, Perl, Python).
- Desarrollos a medida: en el caso de ser necesario realizar algún tipo de desarrollo, se llevará a cabo con tecnologías ampliamente disponibles en cualquier sistema operativo UNIX (p. ej., *shell scripts*, Perl *scripts*, etc.).

ANOTACIONES

Para completar la descripción del sistema, será necesario hacer referencia al conjunto de **estándares y normas** que hay que considerar en la implementación de éste.

Caso práctico

Normas a seguir en el sistema de copias de seguridad de Soluciones Abiertas, S.A.

Las normas y estándares a seguir en la implementación del sistema de copias de seguridad serán las siguientes:

- En cuanto al sistema operativo, se seguirá el proceso documentado como "Instalación de servidores GNU/Linux" de Soluciones Abiertas, S.A.
- El software de copias de seguridad deberá cubrir los estándares implementados en las unidades de cinta utilizadas (p. ej., DLT).
- Los posibles desarrollos a medida seguirán las normas internas de Soluciones Abiertas, S.A., es decir, las recogidas en el documento "Normas de desarrollo de Soluciones Abiertas, S.A.". En este documento quedan recogidas las normas que hay que seguir en el desarrollo de cualquier proyecto, tales como utilización de diagramas UML, formato de documentación del código, etc.

Una vez descrito el sistema, se procederá a **identificar aquellos usuarios que intervendrán en la definición de requisitos de éste y en su aceptación definitiva**. Es especialmente importante contar con la colaboración de los usuarios a lo largo de todo el proceso de desarrollo del sistema.

Caso práctico

Identificación de usuarios del sistema de copias de seguridad de Soluciones Abiertas, S.A.

El personal involucrado en la definición de requisitos y aceptación de la solución final del sistema de copias de seguridad de Soluciones Abiertas, S.A. es:

ANOTACIONES

- Los jefes de cada uno de los departamentos de la empresa ayudarán a establecer parámetros como la periodicidad necesaria de realización de las copias de seguridad según su nivel de importancia, qué elementos se deben copiar, etc.

- Los administradores del sistema serán aquellas personas que utilicen el sistema de copias de seguridad, tanto en la programación de los *backups* como en su recuperación. Serán ellos los que deberán aprobar los requisitos funcionales en cuanto a interfaz de uso, versatilidad de programación, etc. y darán el visto bueno definitivo al sistema.

2.2. Establecimiento de requisitos

El objetivo de esta fase será **completar los requisitos definidos anteriormente**, contando con la información suministrada por los usuarios. En la medida de lo necesario, **se dividirá el sistema en subsistemas** que permitan su estudio por separado, con el fin de facilitar el análisis de éstos.

La comparación de la descripción de cada uno de los requisitos expresados en esta fase del proyecto con el diseño creado con posterioridad nos permitirá verificar la corrección de este último.

El primer paso en el proceso de establecimiento de requisitos será el de **obtener los requisitos a partir de la información suministrada por los usuarios**. Los requisitos recogidos en las reuniones mantenidas con los usuarios elegidos en la fase anterior serán básicamente los siguientes tipos:

- Funcionales (p. ej., mediante el sistema de copias de seguridad se deberá poder recuperar un fichero en concreto sin tener que deshacer todo el *backup*).

- Rendimiento (p. ej., el proceso de recuperación de un fichero de cualquier *backup* de cualquier día de la semana no puede durar más de una hora de principio a fin).

ANOTACIONES

- Seguridad (p. ej., sólo podrán recuperar ficheros del *backup* aquellas personas que estén autorizadas para ello).
- Implantación (p. ej., las copias de seguridad se guardarán en un lugar físicamente seguro).
- Disponibilidad (p. ej., el sistema de copias de seguridad debe ser utilizado al menos una vez a la semana con el fin de comprobar su correcto funcionamiento).

Caso práctico

Requisito periodicidad de *backups* del sistema de copias de seguridad de Soluciones Abiertas, S.A.

Junto con los jefes de departamento de Soluciones Abiertas, S.A., se ha determinado que se deberá hacer una copia diaria de todos y cada uno de los archivos almacenados en el sistema de compartición de ficheros y en el sistema de control de versiones. Asimismo, se realizará una copia también diaria de la información almacenada en la base de datos del sistema de gestión de proyectos. Mediante entrevistas con los administradores del sistema, se ha determinado que dichas copias de seguridad diarias se deben realizar de manera automática a una hora en la que la utilización de los recursos informáticos sea mínima, por ejemplo, de madrugada.

Una vez descritos cada uno de los requisitos, se procederá a la especificación de los **casos de uso** de cada uno de ellos. Los casos de uso, además de la descripción en sí del problema, cubrirán cómo los usuarios interactuarán con el sistema, qué interfaces utilizarán y cómo se tratarán las condiciones de fallo.

Caso práctico

Caso de uso de periodicidad de *backups* del sistema de copias de seguridad de Soluciones Abiertas, S.A.

La periodicidad de las copias de seguridad de los sistemas de almacenamiento y compartición de ficheros y de gestión de proyectos será totalmente independiente de la utilización de éstos (los usuarios no tendrán que llevar a cabo ninguna acción especial).

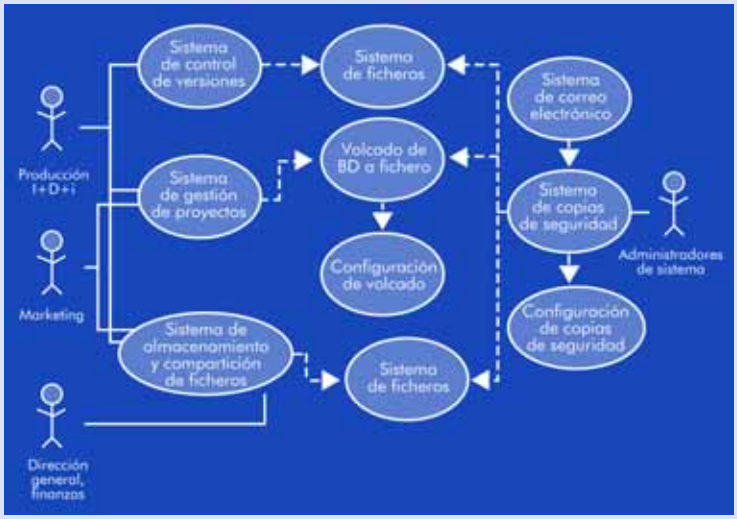
ANOTACIONES

La única condición que se deberá dar para la realización de las copias de seguridad es que los ficheros que se desea copiar estén en los directorios de los cuales se hará copia y que serán previamente determinados (uno de estos directorios contendrá el fichero con el volcado de las base de datos del sistema de gestión de ficheros).

Los administradores del sistema deberán programar la herramienta de copias de seguridad para que se hagan los *backups* con la periodicidad deseada (una vez al día) a la hora determinada (de madrugada). Esta programación se realizará mediante la interfaz de usuario de la herramienta de copias de seguridad, que como ya se ha comentado anteriormente, consistirá en ficheros de texto modificables.

El software de copia de seguridad enviará por correo electrónico un resumen de los resultados de la realización de los *backups*, con lo cual, se advertirá de los posibles errores que se hayan podido dar.

Figura 6. Descripción configuración del sistema de copias de seguridad



A la vez que se describen los requisitos y sus casos de uso correspondientes, se analizarán éstos con el fin de **detectar posibles inconsistencias** (dado que pueden intervenir diferentes usuarios con diferentes necesidades), duplicidades, etc., **y las posibles asociaciones** entre éstos.

ANOTACIONES

Caso práctico

Asociaciones del caso de uso de periodicidad en el sistema de copias de seguridad de Soluciones Abiertas, S.A.

El caso de uso que describe cómo se tratará la periodicidad de los *backups* en el sistema está directamente relacionado con el caso de uso que describe cómo se harán las copias de seguridad del sistema de gestión de proyectos.

La información del sistema de gestión de proyectos estará almacenada en una base de datos de la cual se deberá hacer un volcado a fichero para poder copiarla. Dicho volcado deberá realizarse de acuerdo con la política de periodicidades establecida en el caso de uso "Periodicidad de *backups*". Además, el fichero que contenga el volcado deberá grabarse periódicamente en un directorio del cual se hagan *backups*.

2.3. Definición de interfaces de usuario

En esta fase del análisis **se especifican cómo serán las diferentes interfaces que habrá entre el sistema que estamos describiendo y los usuarios** de éste. Esta especificación se hará teniendo en cuenta los diferentes perfiles de usuarios, flexibilidad necesaria, tipos de acciones que hay que llevar a cabo, etc.

El primer paso en la definición de las interfaces de usuario será el de **definir los perfiles de usuarios que utilizarán el sistema**. De este modo, se podrá describir posteriormente a qué tipos de interfaces accederán cada uno de ellos.

Caso práctico

Perfiles de usuarios de la utilidad de volcado de base de datos del sistema de copias de seguridad de Soluciones Abiertas, S.A.

La utilidad que realice el volcado a fichero de la base de datos del sistema de gestión de proyectos (con el fin de realizar su copia de seguridad) será utilizada por usuarios que cum-

ANOTACIONES

plan las funciones de administradores de sistema, con lo cual tendrán las siguientes características:

- Usuarios con un perfil técnico.
- Usuarios acostumbrados a la utilización de programas sobre terminales, mediante *shells* con interfaces textuales.
- Usuarios acostumbrados a la utilización de ficheros de configuración en formato texto.
- Usuarios acostumbrados a la utilización de herramientas en entornos UNIX.

A continuación, se deberán **especificar los principios generales de la interfaz de usuario**, por ejemplo, si se utilizarán interfaces de texto o gráficas, cómo se mostrarán los mensajes de error, cómo se obtendrá ayuda, etc.

Caso práctico

Principios generales de la interfaz de usuario de la utilidad de volcado de base de datos del sistema de copias de seguridad de Soluciones Abiertas, S.A.

La utilidad que realice el volcado a fichero de la base de datos del sistema de gestión de proyectos (con el fin de realizar su copia de seguridad) tendrá las siguientes características:

- La programación de la utilidad se hará mediante ficheros de configuración en formato texto.
- El fichero de configuración en formato texto seguirá los estándares más habituales utilizados en aplicaciones UNIX clásicas (p. ej., separación de campos con el carácter ":", continuación de líneas con el carácter "\", lista de datos con el carácter ",", etc.).
- Los mensajes de error serán mostrados por pantalla, con un código numérico asociado, salvo aquellos que impliquen que el volcado no se ha podido realizar (en este caso, el mensaje de error se enviará por correo electrónico a una dirección indicada).

ANOTACIONES

- Los ficheros de configuración permitirán la inclusión de líneas con comentarios.
- La ayuda del programa de volcado se obtendrá por medio de los canales habituales en la administración de sistemas (p. ej., utilidad *man*).

Una vez identificadas las características generales de la interfaz de usuario, se pasará a **especificar ésta para cada uno de los casos de uso** definidos en el apartado anterior.

Caso práctico

Interfaz de usuario de la utilidad de volcado de base de datos del sistema de copias de seguridad de Soluciones Abiertas, S.A.

La utilidad de volcado deberá configurarse para que almacene el contenido de cierta base de datos en un determinado fichero. Para ello, habrá que configurar mediante un fichero de texto datos como el usuario y la contraseña a utilizar para acceder a la base de datos, en qué servidor reside, etc.

Ejemplo

Ejemplo de fichero de configuración:

```
# Línea con comentarios.
user = usuario:contraseña
host = projects.example.com
database = projects
e-mail = alberto@cometatech.com,marcg@cometatech.com
file = /var/local/dbDump
compression = true
```

2.4. Especificación del plan de pruebas

Para acabar con la fase de análisis se procederá a realizar la especificación del plan de pruebas, que nos **servirá para establecer si el sistema cumple con los requisitos establecidos por los usuarios**.

ANOTACIONES

Se podrán realizar **pruebas** del sistema **a varios niveles**:

- Pruebas unitarias (p. ej., prueba de conexión a la base de datos del sistema de gestión de proyectos).
- Pruebas de integración (p. ej., prueba de copia de seguridad del fichero producto del volcado de la base de datos del sistema de gestión de proyectos).
- Pruebas de sistema (p. ej., prueba de la copia incremental diaria de los sistemas de almacenamiento de ficheros y del de gestión de proyectos).
- Pruebas de implantación (p. ej., prueba de copia de seguridad sobre la unidad de cintas de la que se dispone).
- Pruebas de aceptación (p. ej., copia completa del sistema que validarán los usuarios correspondientes). Este conjunto de pruebas es crítico, ya que será el que permitirá validar el sistema completo. Además del correcto funcionamiento del sistema, deberá tener en cuenta parámetros como la seguridad, rendimiento, disponibilidad, etc.

Para cada una de las pruebas que hay que realizar, se deberá definir el **alcance** de éstas (p. ej., usuarios implicados en las pruebas, productos de las pruebas, criterios de aceptación de las pruebas, etc.), y los **requisitos en el entorno** de pruebas (hardware necesario, librerías disponibles, configuración de accesos, etc.).

Caso práctico

Prueba de integración del software de copias de seguridad de Soluciones Abiertas, S.A. con la utilidad de volcado de la base de datos del sistema de gestión de proyectos

La prueba tendrá las siguientes características:

- Permitirá a los administradores del sistema comprobar que la copia del fichero del volcado de la base de datos del sistema de gestión de proyectos se ha realizado correctamente.

ANOTACIONES

- Como producto de la prueba, obtendremos un fichero con las diferencias existentes entre el archivo almacenado en la copia de seguridad y el archivo producto del volcado de la base de datos.
- La prueba se dará por correcta cuando el fichero de diferencias entre archivos esté vacío.

Con el fin de poder realizar la prueba, será necesario:

- Disponer de los servidores de copias de seguridad y de sistema de gestión de proyectos.
- Disponer de una base de datos de gestión de proyectos con contenido real.
- Disponer de un acceso remoto a la base de datos del sistema de gestión de proyectos.
- Disponer de una unidad de cinta sobre la cual realizar las copias de seguridad.

ANOTACIONES

3. Diseño del sistema

El objetivo de la fase de **diseño** de un sistema de información es obtener los **modelos y especificaciones** que lo definen a partir del análisis realizado en la fase anterior. Las actividades que llevemos a cabo en esta fase nos permitirán determinar las especificaciones de desarrollo e integración, así como definir el entorno de pruebas e implantación necesarios para su correcto funcionamiento.

Concretamente, los resultados que deberemos obtener en esta fase serán:

- La definición del modelo arquitectónico del sistema. Mediante la identificación de sus componentes, sus interacciones y la ayuda de herramientas de modelado obtendremos un mapa de los subsistemas y recursos que intervienen en todos los procesos.

- Las especificaciones y estándares que se usarán tanto en esta misma fase como durante el desarrollo del sistema.

- La identificación de cada subsistema, sus requisitos de integración, licencia y funcionalidades cubiertas.

- Los casos de uso aplicados de los subsistemas anteriormente identificados, debidamente revisados para reflejar el modelo y especificaciones definidos.

- Los componentes, clases o interfaces que deberemos construir en la fase de desarrollo.

- Los requisitos necesarios para proceder con éxito a la implantación del sistema.

Como vemos, muchos conceptos y especificaciones van a determinarse en esta fase, y aunque algunas metodologías recientes aconsejan mezclarla con la fase de desarrollo en un ciclo combinado de

ANOTACIONES

diseño y construcción iterativo, con el objetivo de obtener resultados pronto o de identificar fallos en el diseño a tiempo, es obvio que todas las decisiones en cuanto a especificaciones, estándares o subsistemas que tomemos aquí van a facilitar todas las tareas futuras.

Especialmente importante es la identificación de los componentes que hay que usar y sus licencias, ya que éstos pueden determinar parte de la funcionalidad, la necesidad de desarrollos internos de comunicación entre subsistemas o el tipo de licencia con que deberemos distribuir (si procede) el resultado de nuestro proyecto.

3.1. Arquitectura

La definición de la **arquitectura** del sistema es el primer paso para **identificar los componentes** del mismo y da lugar a las siguientes fases de diseño en las que profundizaremos en cada uno de ellos. El objetivo es disponer de un conjunto de documentos y diagramas completos y concisos que sean comprensibles para la dirección y a la vez sirvan de base para profundizar en el diseño del sistema.

Antes de realizar el diseño más detallado del sistema, será necesario definir las normas y estándares de diseño y construcción.

Una vez acordados los estándares de diseño, podremos ya profundizar y determinar los subsistemas, repitiendo el mismo proceso que seguimos con la arquitectura general del sistema, esta vez con mayor granularidad en cada uno de ellos.

3.1.1. Definición de niveles de arquitectura

Existen varias formas de ver o entender la arquitectura de un sistema:

- Arquitectura conceptual: su propósito es dirigir la atención sobre los grandes bloques que forman el sistema, sin entrar en detalles, e identificar las relaciones entre dichos bloques. Es muy útil para comunicar a la dirección o a departamentos no técnicos una visión global del sistema.

ANOTACIONES

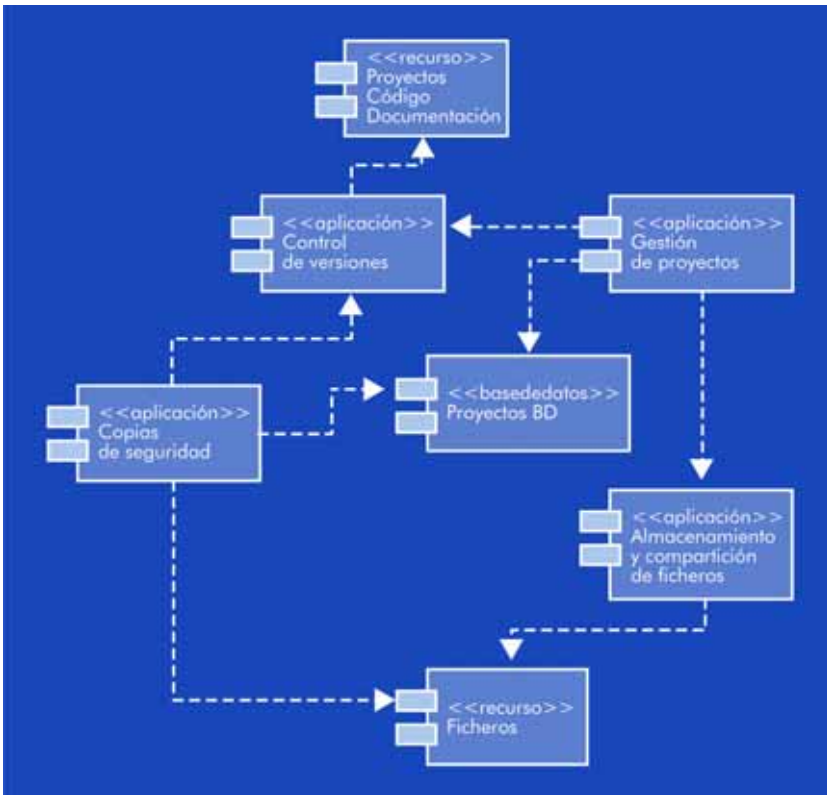
- Arquitectura lógica: añade detalle a la anterior e incorpora la definición de las interfaces de comunicaciones entre los componentes, lo que permitirá a los desarrolladores de cada componente trabajar sin dependencias entre ellos.

Caso práctico

Definición de la arquitectura de la nueva infraestructura software de Soluciones Abiertas, S.A.

Para expresar la arquitectura de la nueva infraestructura, usaremos la notación UML en los diagramas, y usaremos tarjetas CRC (Clase-Responsabilidad-Colaborador) de apoyo.

Figura 7. Diagrama UML de componentes

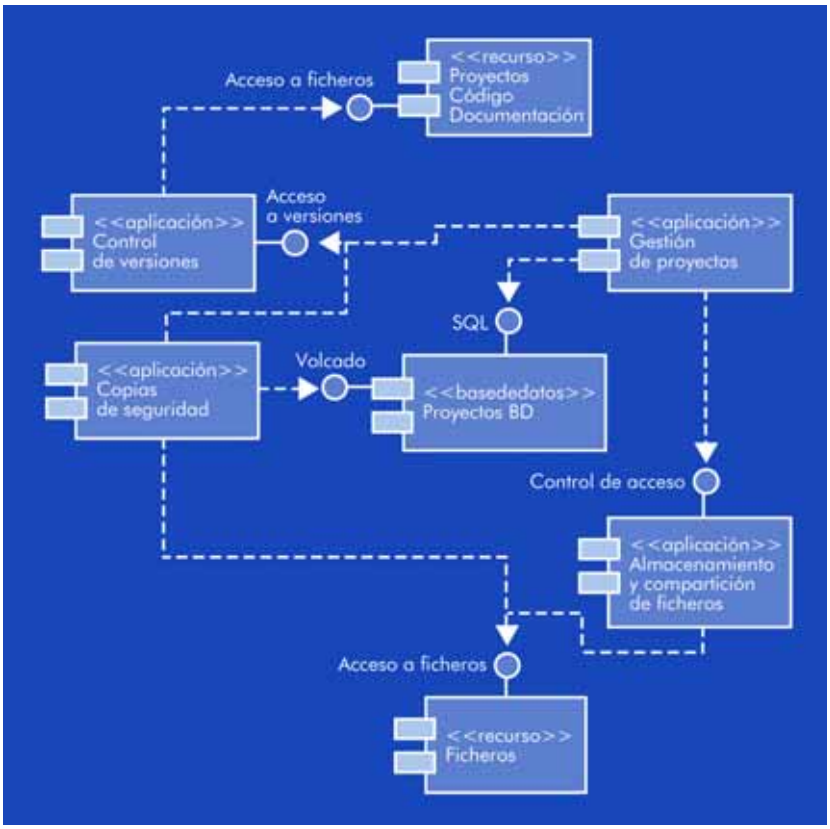


En el diagrama anterior vemos los componentes del sistema y los conectores que los unen. Estos conectores indican que algún tipo de comunicación se produce entre ellos. Es positivo ya mezclar componentes de negocio y componentes técnicos en este diagrama (que se identificarán mediante los estereoptipos <<basededatos>> o <<aplicación>> u otros).

ANOTACIONES

Una vez consensuada esta visión general del sistema, pasaremos a profundizar en las interfaces de los componentes para obtener la arquitectura lógica del sistema. Para ello, extenderemos el diagrama de componentes anterior detallando los procesos de comunicación.

Figura 8. Diagrama UML de componentes con interfaces



Como apoyo a la generación del diagrama anterior o para consensuar las interfaces, podemos utilizar las tarjetas CRC.

Tabla 1.

Control de versiones	
Proporciona versión de trabajo Extrae versiones anteriores y diferencias Almacena versiones y cambios Etiqueta versiones Conoce proyectos	Ficheros

- En la parte superior figura el nombre del componente.

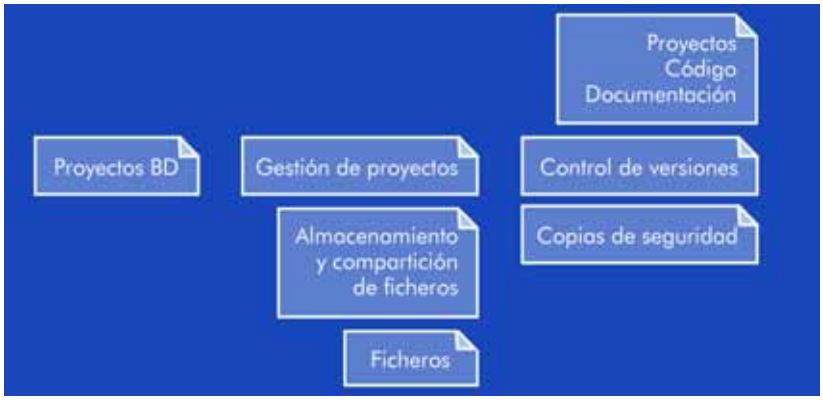
ANOTACIONES

- En la columna izquierda deberemos reflejar todo lo que el componente sabe o hace acerca de él mismo. Allí se incluirá todo aquello que creemos que es de su responsabilidad y la información que deberá mantener.

- En la parte derecha figurarán los componentes con que se relaciona para poder llevar a cabo las responsabilidades de la parte izquierda.

Estas tarjetas son muy utilizadas en las metodologías eXtreme Programming y Agile, y permiten "crear" el diagrama de componentes del sistema dinámicamente encima de una mesa simplemente situando las tarjetas próximas o lejanas entre sí según su grado de comunicación, para consensuar así una visión general de la arquitectura lógica del sistema durante una reunión.

Figura 9. Ejemplo de situación de tarjetas CRC



3.1.2. Especificación de estándares, normas de diseño y construcción

Es importante acordar con las personas encargadas del diseño del sistema y con los equipos que procederán a su construcción unas normas a seguir en la notación de diagramas y documentos. Estas normas pueden venir dadas por estándares o por recomendaciones, o bien ser de uso y creación interna. Obviamente, siempre es recomendable apoyarse en estándares, ya que va a facilitar la comunicación, reusabilidad y comprensión por parte de entidades externas o recién incorporadas al equipo.

ANOTACIONES

Deberemos definir:

- Formato y plantilla de los documentos de diseño.
- Notación a usar en los diagramas de diseño.
- Recomendaciones en cuanto al estilo, idioma y formato de la documentación técnica.

Caso práctico

Definición del conjunto de normas y notaciones de la nueva infraestructura software de Soluciones Abiertas, S.A.

Es conveniente que todos los documentos creados de ahora en adelante y que van a ser objeto de revisión por parte de equipos diferentes compartan unas características y mantengan un formato coherente. Para ello, después de estudiar los estándares y recomendaciones sobre el tema, se llega a las siguientes conclusiones:

- Documentos de diseño: estos documentos se deben poder consultar tanto por el personal técnico implicado, como por personal no técnico que pueda revisarlo o consultarlo. Se acuerda que se trabajen en formato RTF (*rich text format*) y que la versión más reciente esté simultáneamente en PDF para su consulta. Se creará una plantilla que contenga en la primera página:
  - Título del documento.
  - Responsable del documento.
  - Lista de autores que han intervenido y la fecha de su primera intervención.
  - Lista resumida de cambios introducidos en el documento a medida que se vayan produciendo (cambio, fecha y autor).
- Diagramas de diseño: para los diagramas de diseño se acuerda usar la notación Unified Modeling Language – UML (http://www.omg.org/uml/) en su versión 1.5, definida por el Object Management Group (www.omg.org).

ANOTACIONES

- Documentación técnica: la documentación técnica será posiblemente la que más revisiones sufrirá y contendrá también enlaces a documentaciones de las herramientas usadas, especificaciones de programación (APIs), etc., por lo que se recomienda usar un formato lo más flexible posible e integrable con las propias herramientas de desarrollo que se usen. Para ello, se decide usar DocBook (www.docbook.org, http://www.oasis-open.org/docbook/), que nos permitirá:

  – Partición de un documento en varios ficheros estructurados, susceptibles de ser revisados independientemente.
  – Fácil inclusión de referencias a otros documentos (enlaces http, figuras, etc.)
  – Fácil generación de varios formatos para su visualización (PDF, HTML) y con la posibilidad de separar el contenido del documento de su formato.

- Independencia de editor usado, ya que es una implementación de XML y, por lo tanto, modificable en cualquier editor de texto.

- Incorporar documentación contenida en el código fuente generado en la fase de desarrollo, de forma automática en muchos casos.

- Cabe destacar que al tomar las decisiones se ha dado importancia a la implantación del formato o notación en la industria y a la accesibilidad del mismo, es decir, a la disponibilidad de ejemplos y documentación, así como de un amplio conjunto de herramientas que trabajen con ellos.

Identificación de subsistemas

Para reducir la complejidad que supondría diseñar al detalle todo el sistema, éste deberá dividirse en subsistemas para facilitar su comprensión, revisión y reutilización.

ANOTACIONES

Para realizar esta división, podemos tener en cuenta varios aspectos de los componentes:

- Funcionalidad común o relacionada por características de la ejecución.
- Gestión de datos, acceso a datos comunes.
- Integración en una interfaz de usuario común.
- Optimización de líneas de comunicaciones o recursos.

Caso práctico

Identificación y diseño de los subsistemas de la nueva infraestructura software de Soluciones Abiertas, S.A.

Realizando una primera división por funcionalidad, identificamos claramente los siguientes subsistemas:

- Subsistema de copias de seguridad
- Subsistema de gestión de ficheros
- Subsistema de gestión de proyectos
- Subsistema de control de desarrollo y versiones

Si aplicamos el criterio de acceso a datos en la identificación de subsistemas:

- Subsistema de producción. Incluye las funcionalidades de gestión de proyectos, desarrollo y versiones.
- Subsistema de acceso documental. Incluye la funcionalidad de acceso a ficheros compartidos y a la gestión de proyectos con sus documentos relacionados.

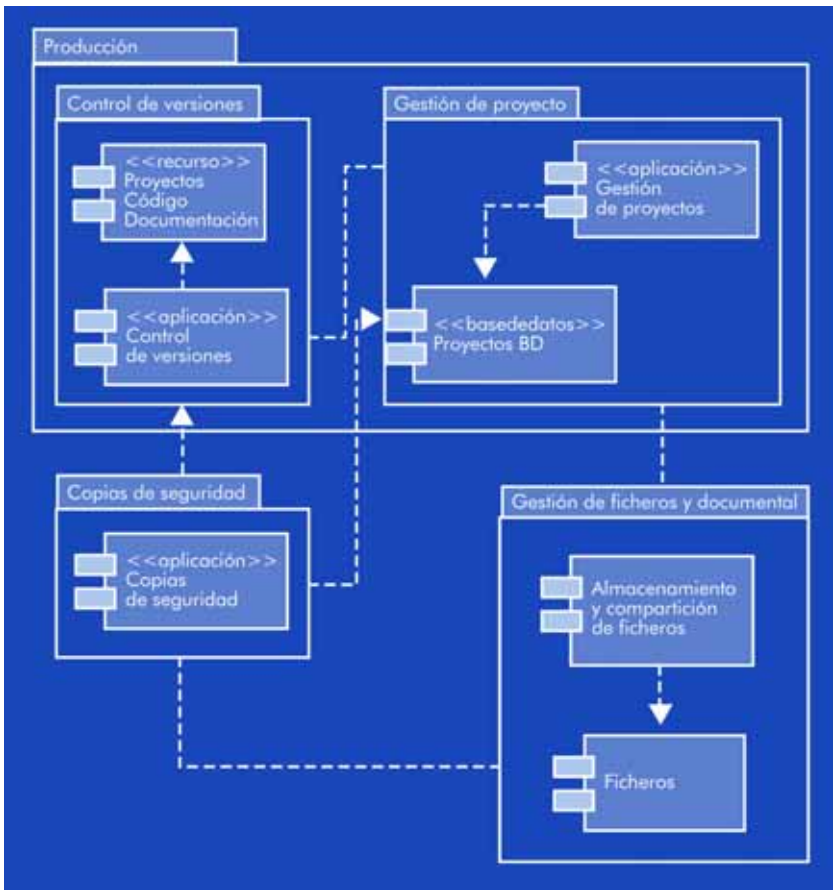
Según este criterio, el subsistema de copias de seguridad no sería relevante, ya que accede a todos los datos sin añadir información ni interpretarlos.

Aplicando el criterio de interfaz común, en este caso sería posible incluir todos los subsistemas vistos en una interfaz integrada y obtener así un único subsistema. Así pues, este criterio no sirve para identificar subsistemas en este caso.

El criterio de optimización de recursos o líneas de comunicación tampoco es aplicable en este caso.

ANOTACIONES

Figura 10. Diagrama UML de componentes



En este diagrama hemos utilizado el diagrama de componentes visto anteriormente y hemos incorporado los distintos subsistemas identificados en forma de paquetes que incluyen uno o varios componentes de la arquitectura del sistema.

Disponemos ahora de un mapa completo de la arquitectura global del sistema. Próximamente, deberemos profundizar en cada subsistema y sus componentes y la mejor forma de hacerlo será mediante la revisión de los casos de uso vistos en el análisis del sistema.

3.2. Casos de uso reales

Una vez identificados los subsistemas, es el turno de revisar los **casos de uso** hechos en la fase de análisis y **determinar las operaciones** que deberán implementar las interfaces de cada uno de ellos.

ANOTACIONES

Así pues, a partir de los escenarios recogidos en la fase de análisis, determinaremos qué subsistemas están implicados en ellos y diseñaremos su funcionamiento teniendo en cuenta:

- El entorno tecnológico en el que se aplican.
- Las excepciones que se produzcan en cada caso de uso.
- Detalles relacionados con la implementación que ya podamos identificar en esta fase.
- Restricciones o características de la interfaz de usuario.
- Nuevos requisitos que podamos identificar.

Si el sistema que estamos diseñando está centrado en el desarrollo, los casos de uso de los subsistemas deberán incorporar la definición de las clases u objetos y por lo tanto los diagramas que obtengamos deberán también representar la interacción entre los mismos.

Así pues, durante esta fase estableceremos las características, revisaremos los requisitos y diseñaremos las clases (con sus atributos, operaciones y relaciones) de todo el sistema. De manera natural durante el proceso, obtendremos también el diseño de las pruebas que asegurarán el buen funcionamiento del sistema durante el desarrollo y las condiciones de implantación del mismo.

3.2.1. Revisión de casos de uso por subsistema

Para cada caso de uso deberemos definir:

- Subsistemas y actores que intervienen en el mismo.
- Mensajes que intercambian los objetos.

La definición de los mensajes nos servirá para verificar y detallar las interfaces de cada subsistema teniendo en cuenta todos los casos de uso en que interviene y así completar la definición de subsistemas realizada en fases anteriores.

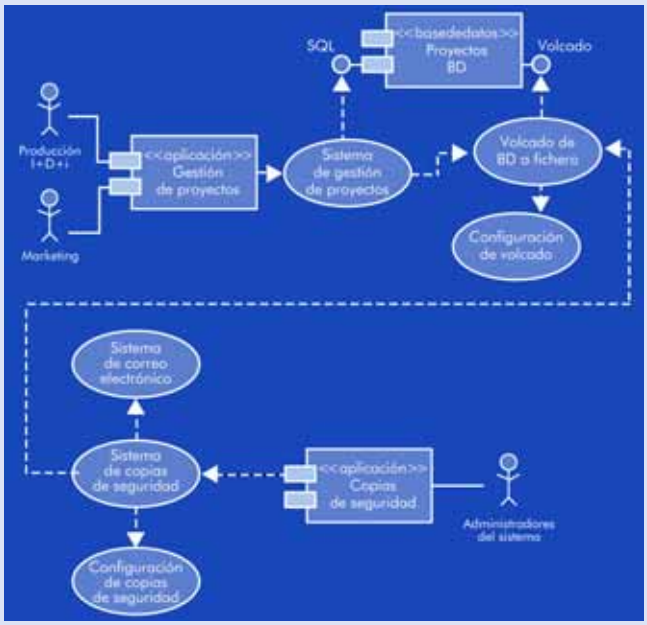
ANOTACIONES

Caso práctico

Revisión de casos de uso del subsistema de copias de seguridad de Soluciones Abiertas, S.A.

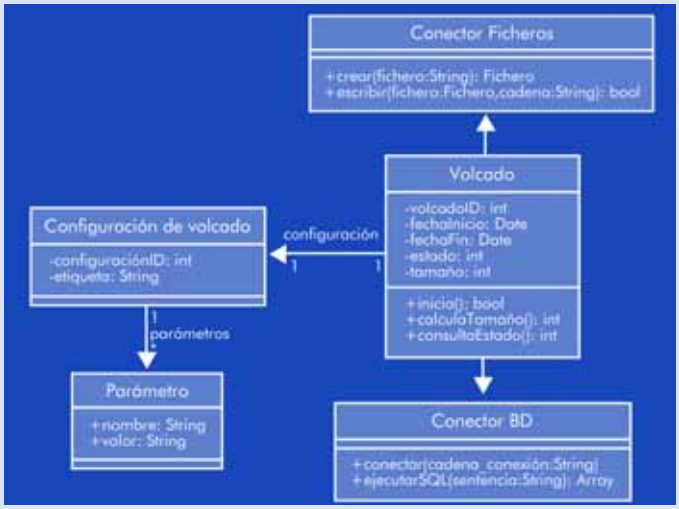
A continuación, completaremos la definición de subsistemas realizada con la revisión de los casos de uso implicados en la gestión de copias de seguridad.

Figura 11. Diagrama UML de revisión de caso de uso con componentes implicados



A continuación, enumeraremos los mensajes que intervienen en el volcado de la base de datos a fichero y los sistemas u objetos con que se intercambian esos mensajes, lo que nos llevará hacia el diagrama de clases del subsistema, donde se representan los objetos que intervienen y sus atributos y métodos.

Figura 12. Diagrama UML de clases



ANOTACIONES

3.2.2. Especificaciones de desarrollo y pruebas

Llegados a este punto, estaremos en condiciones de establecer las condiciones y características del entorno de desarrollo en los siguientes términos:

- Entorno tecnológico: hardware, software y comunicaciones.
- Herramientas de desarrollo: IDEs, generadores de código, compiladores, etc.
- Herramientas de documentación.
- Restricciones técnicas.
- Requisitos de seguridad del entorno.

También deberemos ser capaces de definir las pruebas necesarias que se deberán realizar para asegurar el funcionamiento del sistema una vez implantado. Éstas deberían definirse como pruebas unitarias, es decir, pruebas con el mínimo nivel posible de dependencia entre ellas para permitir un desarrollo o una integración software por componentes, donde cada equipo pueda trabajar y probar independientemente, dejando para la fase final las pruebas de integración.

La especificación de las pruebas unitarias se divide en pruebas de caja blanca y pruebas de caja negra:

- Caja negra: se considera el componente desde el punto de vista funcional, analizando sus entradas y salidas, y comparando todas sus posibilidades con los resultados esperados.
- Caja blanca: se considera el componente como una estructura con una secuencia lógica de eventos y se comprueba la validez de ésta, el código no usado, comprobaciones contempladas, etc.

Normalmente, se usará una combinación de los dos tipos de pruebas, según el componente o su funcionalidad. Tradicionalmente, se tiende a realizar las pruebas de los componentes una vez desarrollados. Las metodologías más recientes recomiendan invertir este proceso, y justifican la realización de las pruebas previamente a las de los propios componentes software por las siguientes razones:

- Al disponer de la funcionalidad requerida de los componentes, estamos en disposición de diseñarlas.

ANOTACIONES

- Desde el primer momento podremos probar que nuestros componentes cumplen o van cumpliendo las funcionalidades.

- Es mejor crear un componente con el objetivo de pasar las pruebas diseñadas, que crear uno que tenga la funcionalidad requerida. De este modo se evita que los programadores introduzcan efectos colaterales o crean que determinada funcionalidad también será necesaria y la introduzcan.

- Se ha demostrado que si las pruebas están bien diseñadas y son completas, los programadores tardarán igual o menos en crear un componente que cumpla las funcionalidades que uno que simplemente pase los tests diseñados.

- Aunque tradicionalmente se dejan las pruebas para el final del desarrollo, éstas son las que más frecuentemente provocan retrasos en el proyecto. Este enfoque de pruebas unitarias mitiga estos retrasos y optimiza el desarrollo que ahora se enfoca directamente a obtener resultados, entendidos como tests pasados satisfactoriamente.

Caso práctico

Definición de las especificaciones de desarrollo y pruebas del subsistema de copias de seguridad de Soluciones Abiertas, S.A.

El subsistema de copias de seguridad debe ser capaz de copiar y restaurar toda la información del resto de subsistemas. Algunos de los recursos donde se almacena la información permiten su acceso y recuperación directa, en un formato inteligible por el sistema de copias. Otros exigirán un desarrollo para proporcionar sus datos en un formato susceptible de ser copiado y restaurado sin afectar a su integridad. Es el caso de la base de datos.

Así pues, se deberá desarrollar:

- Una interfaz que permita el volcado de una imagen de la base de datos en un instante determinado. Esta interfaz deberá poder funcionar a petición de un usuario o bien de forma desatendida cuando se realicen las copias de seguridad periódicamente.

- Una interfaz que permita la sustitución total o parcial de sus datos por otros provenientes de la copia de seguridad.

ANOTACIONES

Al determinar el lenguaje elegido para el desarrollo, se consideran las siguientes alternativas:

- Perl (http://www.perl.org): lenguaje de script muy implantado en tareas administrativas. Muy flexible y versátil, dispone de multitud de librerías de apoyo y una comunidad de usuarios extensa. Permite orientación a objetos.
- Bash (http://www.gnu.org/software/bash/): lenguaje de script sobre el propio intérprete de comandos del sistema operativo. Aunque muy potente, está muy condicionado por el propio entorno de ejecución (variables de entorno del usuario, etc.) y su funcionalidad es menor que la de un lenguaje de programación clásico.
- Python (http://www.python.org): lenguaje de script de uso cada vez más frecuente. Permite una fácil creación de interfaces gráficas, y dispone de multitud de librerías de apoyo. Su intérprete y el propio lenguaje no están tan difundidos como el resto de alternativas.

Por su amplia aceptación y disponibilidad de librerías de apoyo, así como su integración con el sistema operativo, Soluciones Abiertas, S.A. decide utilizar el lenguaje Perl.

Al determinar el entorno de desarrollo, se consideran las siguientes alternativas:

- Jedit (http://www.jedit.org): se trata de un entorno muy potente desarrollado en Java y por tanto multiplataforma, con un conjunto de *plug-ins* que proporcionan funcionalidades adicionales como detección de sintaxis y navegación avanzada por el código, integración con sistemas de control de versiones, etc.
- Vim (http://www.vim.org): se trata de una extensión del editor 'vi' incluido en casi todos los sistemas Unix. Ha sido mejorado para facilitar su uso, y dispone de interfaz gráfica e interfaz de texto.

ANOTACIONES

- Emacs (http://www.gnu.org/software/emacs/): se trata de una herramienta muy potente y compleja. En su origen es un intérprete de Lisp con funcionalidades de edición de textos. Su curva de aprendizaje es muy pronunciada.

Por su facilidad de uso y potencia, con posibilidad de desarrollo de extensiones y de incorporación de las muchas ya existentes, se decide utilizar el entorno de desarrollo Jedit.

El resto de especificaciones de desarrollo provienen de las tomadas anteriormente, ya que el formato de documentación, así como el marco de trabajo de las pruebas unitarias surgen de forma natural a partir del lenguaje de programación escogido.

Marco de trabajo de pruebas unitarias: módulo Perl Test::SimpleUnit

Documentación del propio desarrollo: formato POD (*plain old documentation*)

Documentación técnica de los interfaces y su uso: DocBook.

A continuación, deberemos enumerar las pruebas unitarias, extraídas de las funcionalidades e interfaces del subsistema:

- Conexión y desconexión al gestor de base de datos.
- Obtención de las bases de datos a volcar.
- Conexión y desconexión a cada base de datos a volcar.
- Inicio y fin de transacción con la base de datos para evitar datos corruptos durante el proceso de volcado.
- Obtención de los datos de estructura de cada tabla de cada base de datos.
- Obtención de los datos de cada tabla de cada base de datos.
- Compresión de los datos obtenidos para optimizar recursos.
- Almacenamiento de los datos.
- Obtención de los datos almacenados.

ANOTACIONES

- Descompresión de los datos almacenados.
- Sustitución de la estructura actual por la obtenida de los datos almacenados.
- Sustitución de los datos actuales por los obtenidos de los datos almacenados.
- Comprobación de la integridad de la base de datos.

Para cada una de estas pruebas, deberemos definir sus posibles parámetros o información de entrada, y sus posibles resultados o información de salida. Esto nos permitirá más adelante programar cada uno de los tests bajo el marco de trabajo de tests unitarios elegido.

3.2.3. Requisitos de implantación

Los requisitos de implantación serán los que tenga que cumplir cada componente o subsistema cuando trabaje en el entorno real conjuntamente con el resto de subsistemas. Por entorno no entenderemos únicamente entorno tecnológico, sino que tendremos en cuenta también a los usuarios del subsistema.

Así pues, la implantación del subsistema tendrá implicaciones para los usuarios y habrá que determinar si sus conocimientos actuales son suficientes para usar el nuevo subsistema, o bien deberemos desarrollar un plan de formación.

De la misma manera, desde el punto de vista tecnológico, deberemos determinar las condiciones del entorno donde implantaremos el subsistema, la capacidad actual de sus recursos y sus condiciones de funcionamiento para verificar que nuestro nuevo subsistema no va a agotar los recursos existentes y no va a provocar cambos en los niveles de servicio del resto del sistema.

El documento que recoja los requisitos de implantación deberá contemplar:

- Gestión de la documentación. Quién tendrá acceso a ella y en qué formato y condiciones.
- Formación de los usuarios.

ANOTACIONES

- Necesidades hardware y software.
- Necesidades de comunicaciones.
- Restricciones de rendimiento que tengan lugar en el entorno de implantación.

Este documento se tendrá en cuenta en la fase de implantación, y puede contemplar casos adicionales como la migración del sistema, recuperación frente a desastres, alternativas o posibles ampliaciones de algunos de los recursos más críticos.

A su vez, este conjunto de requerimientos será de gran utilidad para las pruebas de implantación e integración del subsistema en su entorno de funcionamiento.

Caso práctico

Definición de los requisitos de implantación del subsistema de copias de seguridad de Soluciones Abiertas, S.A.

Ya hemos visto que la implantación deberemos realizarla bajo dos enfoques, el del usuario del subsistema y el tecnológico y sus recursos.

Desde el punto de vista del usuario será necesario definir un responsable de la política de copias de seguridad de la empresa. Esta figura tendrá las siguientes responsabilidades:

- Conocer el funcionamiento del subsistema, teniendo acceso a su documentación.
- Diseñar la política de copias de seguridad y consensuarla con los responsables del resto de subsistemas que gestionan los recursos de los que se debe hacer copias de seguridad.
- Comprobar y monitorizar el correcto funcionamiento del subsistema.
- Conocer los posibles riesgos y cómo subsanarlos e incorporar estas actuaciones dentro de la política de copias de seguridad de la empresa.

Debido a la importancia que tiene este subsistema para el funcionamiento de la empresa, y a la urgencia con que suelen pre

ANOTACIONES

sentarse algunos de sus casos de uso, sería conveniente nombrar un segundo responsable que conozca la política de copias de seguridad y los procedimientos necesarios para suplantar al primer responsable si éste no pudiera atender la incidencia.

Desde el punto de vista tecnológico, la implantación del subsistema de copias de seguridad tendrá un impacto sobre todos los recursos de almacenamiento de la empresa.

- El sistema de ficheros que almacena los documentos de la empresa deberá ser accesible por el servicio que realiza las copias de seguridad, sin que el proceso de copia lo modifique.
- El subsistema de control de versiones deberá ser accesible por el servicio que realiza las copias de seguridad, permitiendo el acceso histórico al mismo (a todas las versiones) y a la versión actual de todos los proyectos almacenados.
- La base de datos de proyectos deberá ser accesible por el servicio que realiza las copias de seguridad, y los cambios en su forma de acceso, usuarios y contraseñas deberán ser comunicados al responsable del servicio para evitar interrumpir la prestación del servicio.
- Previamente a cada proceso de copiado, se comprobarán las condiciones iniciales del servicio, tales como:
  - Prevención para evitar que modificaciones simultáneas al proceso de copiado provoquen un fallo en el servicio.
  - Dimensionamiento de recursos para asegurar que el proceso de copiado empieza y termina correctamente.
- Establecimiento de horarios y duración del proceso de copiado para evitar que afecte al funcionamiento normal de la empresa y de la prestación de los servicios del resto de subsistemas.
- Protección del subsistema de copiado para evitar que un error en uno de sus subprocesos afecte al funcionamiento del resto.

ANOTACIONES

De todo este conjunto de requisitos, extraemos las siguientes pruebas unitarias para asegurar la correcta implantación e integración del componente de volcado de la base de datos.

- Obtención del tamaño de la base de datos y del tiempo de copiado de la misma.
- Obtención del tiempo total de restauración de la base de datos.
- Uso simultáneo de la aplicación de gestión de proyectos durante el proceso de copiado.
- Sistema de registro y comunicación de incidencias al responsable.
- Integración del proceso de copiado de la base de datos con el resto de subprocesos de copiado.

ANOTACIONES

4. Desarrollo

El objetivo de la fase de **desarrollo** es la **construcción ordenada del sistema** del que se ha evaluado la viabilidad, analizado y diseñado. El inicio del desarrollo se produce, en metodologías tradicionales, cuando las fases anteriores se han completado satisfactoriamente y en su totalidad. Metodologías más recientes, y en las que se da preferencia a la *agilidad* del ciclo de vida del proyecto, aconsejan pasar a esta fase lo antes posible, avanzando simultáneamente con el análisis del sistema.

No obstante, siempre es conveniente adaptar las metodologías a nuestras necesidades, y siempre es interesante, al igual que en muchos otros ámbitos del software, utilizar los modelos, notaciones o módulos dentro de una metodología muy extensa, que se adapten a nuestra realidad.

Si parte de los desarrolladores deben implicarse en el diseño, puede ser ventajoso para el mismo que se empiece el desarrollo de un subsistema, a la vez que se trabaja en el diseño de otro, y quizá seremos capaces de detectar necesidades de implantación o excepciones que de otro modo provocarían una revisión de parte del diseño con las consecuencias que podría tener al revisar otros subsistemas o componentes.

Existen, sin embargo, un conjunto de especificaciones que conciernen al desarrollo, que deben ser definidas cuando se empiece esta fase, independientemente del ritmo que queramos imprimir al proyecto.

Deberemos ser capaces de planificar el inicio y fin del desarrollo sincronizado de las distintas actividades que al final dejarán al proyecto en condiciones de implantarlo, como son el alcance del propio desarrollo, componentes de terceros a usar o adquirir, sus pruebas, los manuales de usuario, documentación, formación, etc.

ANOTACIONES

4.1. Planificación de las actividades de integración de sistema

En este punto tenemos ya información sobre qué necesita ser desarrollado, qué tipo de componentes de software vamos a integrar en nuestro sistema, qué herramientas vamos a utilizar, en qué entorno, etc.

Las actividades de desarrollo que nos permitirán alcanzar el objetivo planteado son:

- Concretar versiones o alternativas de los componentes de software, servicios o librerías que vayamos a usar.
- Estudiar esos componentes, servicios o librerías.
- Implantar el entorno de desarrollo.
- Desarrollar las pruebas unitarias.
- Desarrollar los componentes necesarios.
- Realizar la documentación.
- Planificar la formación a usuarios del sistema.
- Desarrollar las pruebas de integración del sistema.
- Aprobar el sistema.

El objetivo final de esta fase es la **aprobación del sistema para que pueda ser implantado**. Por lo tanto, todo el resto de actividades debe planificarse hacia el cumplimiento de las condiciones de aprobación del mismo.

El orden de las actividades no tiene por qué ser secuencial, y en su grado de paralelismo tendrá mucho que ver la diversidad de recursos y de perfiles que intervengan en el desarrollo. Aun así, hay actividades que mejoran su despliegue desarrollándose en paralelo (como son el propio desarrollo de componentes, su documentación técnica y sus pruebas unitarias) y otras en las que su inicio depende de resultados obtenidos en otras actividades.

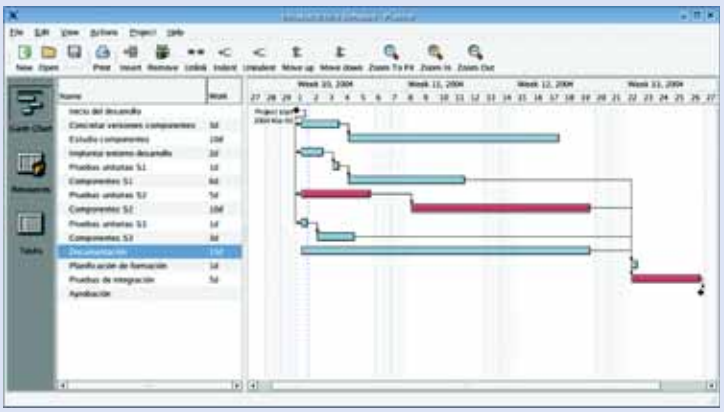
ANOTACIONES

Caso práctico

Planificación del desarrollo de la infraestructura software de Soluciones Abiertas, S.A.

Un método muy usado para expresar este tipo de planificaciones es el diagrama de Gantt. En una primera versión, podemos limitarnos a decidir la duración relativa y la sincronización de las diferentes tareas e hitos.

Figura 13. Planificación de proyecto



A continuación podemos añadir el uso de recursos en cada tarea.

Figura 14. Reporte de planificación completa de proyecto

Estos diagramas, acompañados de sistemas de control de tiempo empleado en tareas (y registrados en hojas de seguimiento de las tareas del proyecto), permitirán a Soluciones Abiertas, S.A. seguir la evolución del desarrollo y ajustar sus recursos, hitos y fechas estimadas según la marcha del mismo.

ANOTACIONES

La plantilla que se usará para reflejar el estado del proyecto periódicamente tendrá este formato:

Tabla 2.

Proyecto	**Infraestructura software de Soluciones Abiertas**
Fecha de este reporte:	DD/MM/AAAA
Fecha de finalización:	DD/MM/AAAA Estimado original DD/MM/AAAA Estimada actual Cambio desde el último reporte +/- DD días
Ítems pendientes de desarrollo	18 defectos 9 funcionalidades
Ítems pendientes de aprobación	0 defectos 2 funcionalidades
Ítems cerrados	10 funcionalidades
Recursos usados durante este periodo	Persona: 8 horas Persona: 23 horas Persona: 5 horas
Resumen del estado	El proyecto está avanzado según lo planificado.
Documentos relacionados	Plan de proyecto. Diseño de módulos.
Estado detallado	
Esta semana nos hemos concentrado en.... Estamos aproximadamente en el 30% del proyecto y llevamos dos días de antelación respecto a la fecha prevista...	
Gestión de riesgos	
Pueden surgir problemas con la versión de uno de los componentes a integrar. Quizá deberíamos evaluar las versiones siguientes o anteriores.	
Actividades planeadas	
Arreglar ítem #452 Arreglar ítem #19 Próxima reunión de desarrollo el DD/MM/AAAA Avanzar en el desarrollo del componente Y	
Actualización de la planificación	
Aquí podemos copiar el diagrama de Gantt visto en la planificación del proyecto, debidamente actualizado según el uso de recursos, desviaciones, etc.	

ANOTACIONES

Caso práctico

Concretar versiones o alternativas de los componentes de la infraestructura software de Soluciones Abiertas, S.A.

Habiendo identificado los componentes necesarios, los concretamos e incorporamos su documentación a la del proyecto.

Tabla 3.

Componente	Paquete	Versión	Licencia
Sistema de copias de seguridad	Amanda	2.4.4p2	Libre distribución con *copyright* de la Universidad de Maryland. BSD.
Sistema de gestión de proyectos	Gforge	3.2	GPL
Sistema de compartición de ficheros	Samba	3.0.2	GPL
Base de datos	PostgreSQL	7.4.1	GPL
Sistema operativo	GNU/Linux	2.4.24	GPL
Servidor web	Apache	2.0.48	Apache Software License 1.1. BSD.

4.2. Elegir la licencia más adecuada

Tanto si se trata de un proyecto interno, como de un producto que tenemos previsto comercializar, **elegir la licencia** del mismo o de las partes que vamos a desarrollar antes del inicio del mismo es altamente aconsejable. Más aún cuando estamos integrando componentes software o servicios, ya que su licencia puede condicionar los términos de la nuestra, u obligarnos a incluirla.

La licencia elegida va a tener algún efecto sobre:

- Los ficheros de código fuente de nuestro desarrollo. Deberemos incluirla en todos y cada uno de ellos, y en determinados casos

ANOTACIONES

hacer mención de partes de código o librerías que son propiedad de otras organizaciones.

- La documentación y los materiales de formación. Ahí debe reflejarse la licencia escogida desde la primera versión del sistema. Esto es especialmente importante en proyectos de código libre que vayan a ponerse a disposición públicamente, para evitar malentendidos.

- Los componentes elegidos. Si el desarrollo va a ser comercializado bajo una licencia propietaria, debemos asegurarnos de que los componentes que integremos nos lo permiten.

- El cliente que recibe el producto. Debemos informar al cliente respecto a qué derechos tiene sobre el producto y qué garantías le proporcionan sus derechos.

- El mantenimiento del mismo, y el soporte que vamos a ser capaces de proporcionar al producto. En la fase de implantación veremos las estrategias de mantenimiento posibles según el tipo de licencia escogido.

Básicamente debemos concentrarnos en las incompatibilidades entre los distintos modelos de licencias existentes, dentro de alguno de estos escenarios:

- ¿Vamos a combinar código propietario con el nuestro y distribuirlo bajo licencia comercial? ¿Hemos adquirido estos derechos por parte de los distribuidores del código propietario?

- ¿Vamos a combinar código libre (en alguna de sus variantes, BSD, GPL, LPGL, etc.) con el nuestro y distribuirlo bajo licencia comercial? ¿Nos lo permiten todas las licencias de los distintos componentes?

- ¿Vamos a combinar código libre con el nuestro y distribuirlo bajo licencia libre? ¿Cuál deberá ser la licencia libre resultante? ¿Deseamos imponer alguna restricción a nuestra licencia libre? ¿Nos interesa mantener el *copyright*? ¿Dar garantía y soporte?

En todo caso, la respuesta a las preguntas que surgen en cada escenario deberán resolverse después de una lectura detenida de las li-

ANOTACIONES

cencias de los componentes y de un análisis de nuestro modelo de negocio.

Caso práctico

Elección de la licencia del desarrollo de la infraestructura software de Soluciones Abiertas, S.A.

El caso práctico planteado es un desarrollo interno, por lo que la licencia elegida no tendrá efectos sobre el modelo de negocio ni sobre su distribución a clientes. Esto no evita que debamos incluir una licencia en el código que desarrollemos, y en este caso, tenemos las siguientes alternativas:

- Licencia propietaria: en nuestro sistema estamos combinando licencias estilo BSD y GPL. Si no vamos a redistribuir nuestro sistema, podemos desarrollarlo bajo licencia propietaria.

- Licencia estilo BSD: esta licencia nos permite mantener el *copyright* sobre nuestro desarrollo, y es coherente con las licencias del resto de componentes. No nos obliga a distribuir el código fuente resultante, pero sí que permite al destinatario del software su uso, copia, modificación, redistribución o venta del mismo. También nos permitirá su incorporación futura a un producto comercializable bajo una licencia propietaria.

- Licencia GPL: esta licencia nos permite mantener el *copyright* sobre nuestro desarrollo, y es coherente con las licencias del resto de componentes. Nos obliga a distribuir el código fuente resultante e impide su futura comercialización bajo una licencia propietaria.

Una vez analizadas las alternativas, decidimos adoptar la licencia BSD, ya que deja abierto el sistema a nuevas incorporaciones de componentes y a su futura comercialización bajo los términos que consideremos oportunos en ese momento.

ANOTACIONES

4.3. Entorno de desarrollo

El objetivo de esta fase es **implantar el entorno de desarrollo seleccionado** en el equipo que lo va a llevar a cabo y proceder con el mismo. Entrando en detalle sobre las actividades vistas en la planificación del desarrollo, identificamos las siguientes tareas:

- Preparación del entorno de generación y desarrollo.
- Generación del código de los componentes o procedimientos.
- Ejecución de las pruebas unitarias.
- Ejecución de las pruebas de integración.

Si se han seguido las recomendaciones en cuanto a metodología de análisis y diseño en las fases anteriores, y se dispone de herramientas de asistencia al desarrollo o de generación de código a partir de los diagramas de componentes o clases, el equipo de desarrolladores dispone de toda la información y herramientas necesarias para terminar con éxito su participación en el proyecto.

Si hemos implicado a los desarrolladores en actividades de diseño (como sugieren algunas metodologías recientes) el tiempo de estudio del mismo y de los requisitos del sistema se verá reducido, imprimiendo un mayor ritmo al proyecto en el inicio del desarrollo. Por contra, si se han dejado fases del diseño pendientes de aprobación por haber empezado el desarrollo de otros subsistemas completamente analizados, tenemos que incorporar estas actividades que interrumpirán el desarrollo dentro de la planificación.

Los resultados de las pruebas unitarias diseñadas y desarrolladas anteriormente son un perfecto testigo del avance del proyecto, y nos permitirán conocer en todo momento si el ritmo del desarrollo es el deseado o nos estamos desviando de nuestra planificación.

ANOTACIONES

Caso práctico

Entorno de desarrollo del software de volcado de la base de datos para el subsistema de copias de seguridad de Soluciones Abiertas, S.A.

Según las decisiones tomadas en las fases anteriores, estamos en condiciones de llevar a cabo el desarrollo de la herramienta de volcado de la base de datos. A partir de la planificación del desarrollo, debemos:

- Instalar el IDE Jedit en los ordenadores de los desarrolladores.
- Acordar un estilo de codificación. Elegimos basarnos en el definido por el autor del propio lenguaje: http://www.perldoc.com/perl5.6/pod/perlstyle.html
- Acordar un conjunto de preferencias del funcionamiento del editor, tamaño de tabulación, estilo de código, etc., de acuerdo al estilo de codificación acordado.
- Generación de código a partir de los diagramas de clases, los casos de uso revisados y las pruebas unitarias.
- Ejecución de pruebas unitarias concurrentemente con el desarrollo.

4.4. Documentación

El objetivo de esta fase es la **elaboración de la documentación de usuario**. Sobre la base de las decisiones tomadas al respecto en fases anteriores, relativas a su formato y disponibilidad, debe desarrollarse su estructura y contenido.

Si se trata de una documentación técnica, deberemos poder incorporar documentación de otras fuentes o componentes que integremos en el sistema. De igual manera, deberemos poder incorporar la documentación de las operaciones y procedimientos que se encuentren en el código fuente.

ANOTACIONES

En el caso de incorporar referencias externas, es de vital importancia citar explícitamente la versión que se ha integrado en el sistema, ya que los enlaces externos pueden hacer referencia a la última versión (que puede coincidir con la integrada en el momento de la creación de la documentación), y más adelante podrían producirse discrepancias.

El estilo de redacción de la documentación tiene que ser acorde con el destinatario final de la misma, y se deben indicar claramente los cambios producidos desde la versión anterior.

Caso práctico

Documentación del software de volcado de la base de datos para el subsistema de copias de seguridad de Soluciones Abiertas, S.A.

Las decisiones tomadas en fase anteriores indican que la documentación debe estar en formato DocBook.

La documentación de las librerías usadas para conexión a la base de datos, tests unitarios, etc. se encuentra en formato POD, así como la documentación técnica contenida en el código fuente.

Mediante la utilidad Pod-DocBook (http://search.cpan.org/~nandu/Pod-DocBook-1.0/) podremos volcar la documentación de librerías y de nuestro propio código en la documentación.

Otros componentes o librerías que no dispongan de utilidades similares, podrán incorporarse a la documentación como enlaces hipertexto.

ANOTACIONES

5. Implantación

El **paso a producción del sistema** en el entorno en que va a operar requerirá una cuidada planificación de sus actividades. En los casos en que el sistema sustituya a otro de similar funcionalidad, nos encontraremos con un tipo de escenarios determinado, y en los casos en que el sistema que se desea implantar añada prestaciones o funcionalidad a un sistema ya implantado, la incidencia de determinadas actividades será distinta.

En todo caso, es en esta fase donde deberemos implicar a los usuarios participantes de los casos de uso analizados del sistema y formarlos en sus nuevas responsabilidades o cometidos.

También deberemos acordar el nivel de servicio que debe prestar el sistema, según las especificaciones, y comprobar que éste se cumple. En el caso en que el sistema deba integrarse en uno ya existente, verificar que su implantación no afecta a su funcionamiento será una de las actividades clave.

Caso práctico

Planificación de la implantación de la nueva infraestructura software de Soluciones Abiertas, S.A.

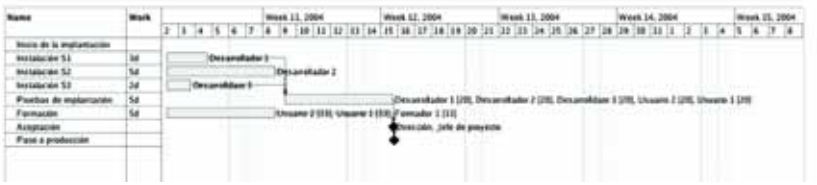
El calendario de la implantación deberá realizarse atendiendo a las distintas fases implicadas. Para su representación y seguimiento, pueden usarse el mismo tipo de herramientas usadas en la planificación del desarrollo.

Al implantar todo el sistema, los equipos de trabajo y usuarios del resto de subsistemas deberán implicarse en su testeo, y por lo tanto es muy importante disponer de herramientas colaborativas de comunicación y registro de incidencias. El mismo gestor de proyectos puede servirnos para registrar los problemas o incidencias que se vayan produciendo y su solución.

ANOTACIONES

El diagrama de Gantt correspondiente a la planificación de la implantación del sistema podría ser el siguiente:

Figura 15. Diagrama de Gantt para la planificación de la implantación



5.1. Formación

La formación en sí misma no suele formar parte de las metodologías relacionadas con proyectos de sistemas de información. Lo que es importante en esta fase es determinar los distintos perfiles de usuario que van a necesitar ser formados sobre el sistema que estamos implantando y adaptar la misma a esos perfiles. El formato y tipo de formación (presencial, no presencial, etc.) también puede venir determinado por esos perfiles de usuario, como puede ser el caso de trabajadores que ejerzan su actividad fuera de las instalaciones de la empresa u otras organizaciones que deban usar el sistema implantado.

En cualquier caso, los estudios de los casos de uso realizados en las fases de análisis y diseño del sistema serán la base para definir el plan y contenidos de la formación.

5.2. Implantación del sistema y pruebas

En esta etapa se llevarán a cabo las **actuaciones para implantar el sistema en su entorno operativo definitivo**. Mientras que las pruebas unitarias y de integración tienen lugar en un entorno diferente del entorno de operación real, las **pruebas de implantación** deben realizarse en el sistema en producción.

ANOTACIONES

Por tanto, se deberá verificar como paso previo que los recursos necesarios atendiendo a los requisitos especificados están disponibles, así como servicios auxiliares, bases de datos, etc.

Adicionalmente, es posible que se deban cargar datos iniciales en el sistema. Esta tarea deberá ser contemplada en esta fase, e incorporada a las pruebas de implantación con los casos reales.

Caso práctico

Implantación y pruebas de la nueva infraestructura software de Soluciones Abiertas, S.A.

Con todos los procedimientos documentados y el sistema probado a nivel de integración, la implantación se limita en este caso a realizar la instalación de los servicios en los servidores definitivos y a implantar las políticas de acceso a dichos servicios, junto con los datos que se especificaron en los casos de uso.

Así pues, de esta actividad debemos:

- Instalar los servicios en los servidores designados.
- Instalar los componentes desarrollados en los servidores designados.
- Instalar el gestor de base de datos, definiendo sus usuarios y políticas de acceso.
- Crear las bases de datos, la estructura de sus tablas y cargar sus datos iniciales.
- Configurar los servicios externos que vayan a usarse por parte de los servicios, como por ejemplo el servidor de correo saliente.

Una vez comprobada la correcta instalación del sistema, activaremos las tareas periódicas que se ejecuten desatendidas (como las copias de seguridad) monitorizándolas durante un ciclo completo de su periodicidad.

ANOTACIONES

Caso práctico

A continuación ejecutaremos las pruebas de implantación, de acuerdo a las especificaciones establecidas. Entre otras, podrían ser:

- Recuperación del sistema forzando un comportamiento erróneo de algunos de los subsistemas. Debemos registrar las operaciones que se han seguido y el estado de los datos y los servicios después de la recuperación.
- Rendimiento del sistema en el entorno de operación. Tiempo de respuesta, comunicaciones, etc.

Una vez revisados los resultados y contrastados con los requisitos, habrá que decidir si hay incidencias que solventar y qué equipo o subsistema es el responsable. Posteriormente, y según la gravedad de las desviaciones o incidencias registradas, decidiremos si volvemos a ejecutar el plan de pruebas completa o parcialmente.

5.3. Nivel de servicio

Según los resultados obtenidos en las pruebas de implantación, y los requisitos del sistema, estaremos en condiciones de **fijar un nivel de servicio** de cada subsistema. El acuerdo de nivel de servicio deberá contemplar:

- Identificación de los servicios en los que va a haber acuerdo. De qué tipo de servicios estamos hablando:
  - Soporte en línea: tiempo de respuesta, disponibilidad.
  - Comunicaciones.
  - Seguridad.
  - Gestión de recursos: capacidad, horas de servicio para el usuario.
- Propiedades de cada servicio. Unidades en las que se mida la prestación del servicio. Pueden ser numéricas (horas de tiempo de respuesta, capacidad de disco para los usuarios, ancho de banda, etc.) o bien expresadas en términos de restricción de las capacidades del sistema cuando hablemos de seguridad, por ejemplo.
- Estimación de los recursos empleados en prestar el acuerdo de nivel de servicio según los compromisos acordados.

ANOTACIONES

5.4. Aceptación del sistema

Esta actividad consiste en **presentar a los responsables o a dirección toda la documentación relativa a la implantación del sistema**, incluyendo los resultados de las pruebas y el acuerdo de nivel de servicio, **para su aprobación**.

Sólo cuando el sistema ha sido presentado y aprobado formalmente, se considera su paso a producción, y por tanto empieza la prestación del nivel de servicio acordado y su mantenimiento.

ANOTACIONES

6. Mantenimiento

Aunque hemos situado esta fase después de la aprobación del sistema (ya que es cuando se inician sus actividades), su planificación se debe producir a lo largo de todas las fases del proyecto. Muchas de las actividades ya realizadas, como la documentación o el establecimiento de requisitos de desarrollo, van orientadas a facilitar también el **mantenimiento del sistema**.

A diferencia de los proyectos en los que usamos software propietario, donde estamos obligados a contratar a la organización propietaria del software o alguno de sus socios, el mantenimiento en proyectos de software libre admite mucha más flexibilidad. Nos podemos encontrar con todos o algunos de los siguientes escenarios:

- Posibilidad de contratar un soporte técnico del componente de software libre a la empresa desarrolladora.

- Posibilidad de formar a nuestros desarrolladores en el componente, ya que disponemos del código fuente, podemos asegurar que si poseen los suficientes conocimientos técnicos podrán solventar la mayoría de incidencias que se produzcan. Si esto sucede, podemos proporcionar la solución de la incidencia a los desarrolladores del producto para que la incorporen. Durante este proceso, normalmente no tendremos ninguna dificultad en conseguir ayuda por parte de los desarrolladores del componente.

- Posibilidad de contratar a una tercera empresa, experta en la herramienta, que nos proporcione el soporte necesario, si la empresa original no ofrece este servicio o el tiempo de respuesta a nuestra petición no se ajusta a nuestro calendario.

Dependiendo de nuestra capacidad técnica, de nuestros recursos y de la urgencia con que necesitemos solventar la incidencia o implementar la nueva funcionalidad, optaremos por una fórmula u otra.

ANOTACIONES

Caso práctico

Mantenimiento del subsistema de copias de seguridad de Soluciones Abiertas, S.A.

Durante la última realización de las copias de seguridad, hemos encontrado un problema en el volcado de la base de datos. Como el software Amanda escogido para hacer las copias en cinta no tiene en cuenta el tamaño de la misma antes de hacer la copia, el proceso no terminó correctamente.

Nuestros desarrolladores nos indican que la única solución sería modificar el código fuente de Amanda para que no empezara el volcado si los ficheros a copiar en cinta exceden de un determinado tamaño. De este modo se podría avisar al administrador y la copia se realizaría correctamente después de solventar la incidencia.

Consultando el sitio web del producto, vemos que existe una sección donde es posible contactar con empresas que dan soporte al mismo. Contactamos con varias de ellas que nos mandan diferentes presupuestos y después de decidirnos por la que nos ofrece mejores condiciones en cuanto a tiempo de respuesta, recibimos el parche para el producto que pasamos a incorporar en un entorno seguro.

A continuación, ejecutamos las pruebas de integración, y posteriormente implantamos y probamos el subsistema implicado.

ANOTACIONES

Resumen

A lo largo del material del curso se han podido repasar las diferentes **fases de las que consta un proyecto** de sistemas de información. Dichas fases, dependiendo de la metodología utilizada (especialmente si forma parte del grupo de las denominadas clásicas, o por el contrario forma parte de las nuevas metodologías ágiles) pueden hacerse presentes antes o después dentro de la vida del proyecto, desarrollarse de manera secuencial o en paralelo, etc.

Las fases básicas en las que se suelen dividir los proyectos de sistemas de información son las siguientes:

- **Estudio de viabilidad**: en esta fase se considerará si el proyecto se puede llevar a cabo, teniendo en cuenta las diferentes soluciones existentes y los recursos de los cuales se dispone.

- **Análisis**: en esta fase se estudiarán las necesidades que se desea satisfacer con el nuevo proyecto, con el fin de poder enfocar la solución tecnológica. Asmismo, se especificarán las interfaces de usuario que permitirán a éstos interactuar con el sistema.

- **Diseño**: en esta fase se realizará el diseño tecnológico de la solución, proponiendo una arquitectura global de ésta y estudiando cada uno de los casos de uso existentes.

- **Desarrollo**: en esta fase se construirá la solución, teniendo en cuenta temas como el entorno de desarrollo utilizado, licencias utilizadas, documentación generada, etc.

- **Implantación**: en esta fase se pasará a un entorno de producción la solución desarrollada, realizándose la aceptación definitiva de éste.

- **Mantenimiento**: durante esta fase, que se prolongará a lo largo del resto de la vida del proyecto, se realizará el mantenimiento de éste, tanto a nivel correctivo como evolutivo.

ANOTACIONES

En caso de tratarse de un proyecto a desarrollar dentro de un marco de tecnologías de software libre, en cada una de las fases anteriores se deberá tener en cuenta ciertos aspectos que, si bien normalmente ya se tienen en cuenta, en este tipo de entorno adquieren una especial relevancia (p. ej., viabilidad de la solución, arquitectura global del sistema, licencias utilizadas).

ANOTACIONES

Bibliografía

Métrica 3 (http://www.csi.map.es/csi/metrica3). "Consejo Superior de Informática y para el impulso de la Administración Electrónica". Ministerio de Administraciones Públicas.

Dia a drawing program (http://www.lysator.liu.se/~alla/dia/).

ReadySet (http://readyset.tigris.org/)

Planner (http://planner.imendio.org/)

AgileAlliance (http://www.agilealliance.org/)

Extreme Programming http://www.extremeprogramming.org/

Free License Quick Reference http://zooko.com/license_quick_ref.html

The Object Management Group – UML (http://www.omg.org/uml/)

ANOTACIONES

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

ANOTACIONES

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

ANOTACIONES

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

ANOTACIONES

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

ANOTACIONES

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

ANOTACIONES

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

ANOTACIONES

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

ANOTACIONES

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

ANOTACIONES

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the

ANOTACIONES

translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

ANOTACIONES


UOC
www.uoc.edu