PROYECTO DE FINAL DE CARRERA EN LA INGENIERÍA TÉCNICA EN INFORMÁTICA: GUÍA DE REALIZACIÓN Y DOCUMENTACIÓN

Versión 1.52 – 18 de marzo de 2000

Francisco José García Peñalvo
Departamento de Informática y Automática
Universidad de Salamanca
fgarcia@gugu.usal.es

Jesús Manuel Maudes Raedo
Departamento de Ingeniería Civil
Universidad de Burgos
jmaudes@ubu.es

Mario Gerardo Piattini Velthuis
Departamento de Informática
Universidad de Castilla-La Mancha
mpiattin@inf-cr.uclm.es

José Rafael García-Bermejo Giner
Departamento de Informática y Automática
Universidad de Salamanca
coti@gugu.usal.es

María N. Moreno García
Departamento de Informática y Automática
Universidad de Salamanca
mmg@gugu.usal.es

Colaboran:

Departamento de Informática y Automática
Universidad de Salamanca

Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos

Departamento de Informática
Universidad de Castilla-La Mancha

Lista de cambios

Versión	Fecha	Descripción
1.0	15/05/1999	Primera versión distribuida internamente en el Departamento de Informática y Automática de la Universidad de Salamanca.
1.1	27/06/1999	Introducción de los comentarios realizados en el Departamento de Informática y Automática de la Universidad de Salamanca.
1.2	12/07/1999	Primera versión distribuida a los alumnos de la Ingeniería Técnica en Informática de Sistemas de la Universidad de Salamanca.
1.3	29/07/1999	Introducción de los cambios propuestos en colaboración con el Área de Lenguajes y Sistemas Informáticos de la Universidad de Burgos.
1.31	03/08/1999	Correcciones ortográficas.
1.4	21/08/1999	Cambios de redacción, estilo y orden de algunos apartados del texto.
1.5	08/09/1999	Orientación del documento a las Ingenierías Técnicas en Informática. Introducción de cambios propuestos desde el Departamento de Informática de la Universidad de Castilla-La Mancha.
1.51	13/09/1999	Corrección de las erratas de la versión 1.5
1.52	19/03/2000	Actualización de la referencia al estándar IEEE Std 830-1998 – *Recommended Practice for Software Requirements Specifications*

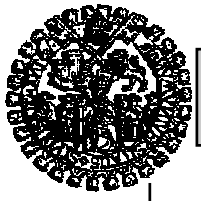
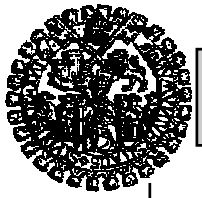


Tabla de contenidos



1. Introducción

La realización del proyecto de final de carrera se convierte en una de las primeras oportunidades con las que cuenta un estudiante de una Ingeniería Técnica en Informática para aplicar de forma integrada los conocimientos teóricos y prácticos obtenidos en el conjunto de asignaturas cursadas en la carrera.

En aquellos casos en dicho proyecto tenga como fin obtener un producto software que resuelva unas especificaciones de requisitos concretas, el producto debe verse acompañado por todo el conjunto de documentos técnicos que se obtienen como resultado de aplicar un proceso metodológico y sistemático en el desarrollo del proyecto.

Es por tanto recomendable que el tutor del proyecto incida en dos aspectos fundamentales: *la aplicación de un proceso metodológico para la realización del producto software* y *el reflejo de dicho proceso en la memoria del proyecto*.

Este documento tiene como objetivos ofrecer unas guías básicas de referencia sobre el proceso de desarrollo a seguir y sobre el formato de la memoria a entregar.

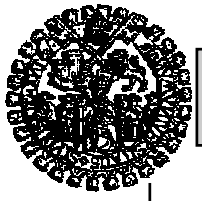
2. El proceso de desarrollo

El proceso de desarrollo del software a seguir va a variar considerablemente dependiendo del soporte metodológico que se elija para el desarrollo, el cual va a estar íntimamente relacionado con el tipo de software que se esté desarrollando.

El marco metodológico se va a derivar del paradigma en el que se quiera desarrollar el producto software, cayendo la mayor parte de los casos bajo el paradigma clásico estructurado (*procedimental*) o bajo el paradigma objetual.

En cualquiera de los casos, y prácticamente con independencia de la metodología de desarrollo que se siga, se van a tener tres fases[1] principales por las que todo proyecto debe pasar: **la fase de análisis**, donde se van a establecer los requisitos que debe cumplir el producto software y se va a elaborar un conjunto de modelos iniciales del sistema que capturen la semántica y comportamiento de éste; **la fase de diseño,** en la que se van a transformar o refinar los modelos de la fase de análisis en otra serie de modelos que vayan perfilando una solución más cercana al mundo de los ordenadores. La fase de diseño va a poner especial énfasis en establecer la arquitectura del sistema software, esto es, obtener la jerarquía de módulos y la estructura de datos del sistema software, aunque también recaen en esta fase tareas tan importantes como son el diseño de la interfaz de usuario, el diseño de los algoritmos y el diseño detallado de los módulos; por último **la fase de implementación,** donde finalmente se van a traducir los diseños de los módulos al lenguaje o lenguajes de programación elegidos para dar forma al sistema final.

[1] Sin olvidarse de la existencia de otras importantes fases como la de definición del sistema o la de pruebas.



Un sistema software de calidad no puede prescindir de ninguna de las fases anteriores.

La fase de análisis debe marcar los problemas a abordar con el sistema software, y por lo tanto es de vital importancia como guía de todo el desarrollo y también como referencia de acuerdo entre la parte que requiere la aplicación y la parte que la desarrolla.

La fase de diseño es la responsable de muchos de los factores que determinan la calidad del software tanto en el ámbito interno como en el externo.

La fase de implementación debe aprovechar las diferentes facilidades que ofrezcan los entornos de programación en los que se codifiquen los sistemas.

Por tanto, es necesario ver al software como un conjunto de elementos que evolucionan, dentro de un proceso de desarrollo, desde un nivel de alta abstracción hasta un nivel de implementación, y no de forma contraria; de manera que la documentación técnica se irá obteniendo a lo largo de todo el proyecto y no después de haber terminado el mismo.

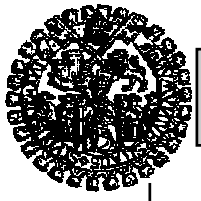
2.1 Proceso de desarrollo de software estructurado

Actualmente la realización de proyectos bajo el prisma de la orientación a objeto es cada vez más frecuente. Sin embargo, el desarrollo de software bajo las directrices de un enfoque procedimental no se puede considerar erradicado, siendo especialmente adecuado para el mantenimiento de sistemas legados, situaciones donde prima la componente funcional y de gestión de datos, o incluso en el software de control.

El soporte metodológico a este tipo de desarrollos puede encontrarse principalmente en el método de Yourdon [Yourdon Inc., 1993] o en la metodología Métrica 2.1 [MAP, 1995].

Las metodologías deben tomarse como una guía del desarrollo, la cual vaya marcando los hitos a conseguir, así como los tipos de documentos, los cuales dan lugar a la documentación técnica del proyecto. No obstante, esto no significa que haya que seguir la metodología hasta sus últimas consecuencias, sino que debe aplicarse y adaptarse de *forma inteligente* al tipo de proyecto que se está desarrollando (*especialmente cuando los proyectos que se están llevando a cabo no son de una gran entidad, como suele ocurrir en los proyectos de final de carrera*).

Normalmente, se ha de buscar que el ciclo de vida del proyecto no caiga en la secuencialidad irreal propugnada por el ciclo de vida clásico [Pressman, 1998], [Piattini et al., 1996], caminando más hacia las iteraciones entre actividades propias del ciclo de vida estructurado [Yourdon, 1993], las cuales se pueden completar con la incorporación de prototipos, ya sea de una forma más tradicional o derivando en un ciclo de vida en espiral [Pressman, 1998], [Piattini et al., 1996], donde las iteraciones se pueden combinar con incrementos y prototipos robustos que evolucionan hacia el sistema final.



2.2 Proceso de desarrollo de software orientado a objetos

La aproximación objetual al desarrollo de sistemas software es una necesidad que se deriva de la creciente complejidad de éstos, así como de la presencia cada vez más habitual de arquitecturas software estructuradas en diferentes niveles o capas para permitir un acceso flexible, versátil y distribuido, en el que se combinan todo tipo de tecnologías, y donde la web aparece con más frecuencia como soporte fundamental del proyecto.

Cualquier entorno de desarrollo actual, que no quiera verse relegado en el mercado, debe hacer gala de orientado a objeto, visual, y con amplias características gráficas, multimedia y de soporte para el desarrollo de aplicaciones web. Lo cual le lleva a incluir unas potentes, completas y complejas APIs (*Application Programming Interface*) basadas en objetos y componentes, que pueden llevar a pensar que cualquier programa que las utilice ya es orientado a objeto, lo cual está bastante alejado de la realidad.

Un desarrollo para que se considere orientado a objeto debe plantearse dentro de la filosofía marcada por el paradigma objetual, esto es, el sistema software es un conjunto de objetos que intercambian mensajes (*o se encuentran relacionados*) para conseguir un objetivo final. Dentro de esta sociedad de objetos aparecen los objetos anteriormente mencionados, y que podrían denominarse objetos de utilidad y de interfaz, pero también deben aparecer los objetos que se denominan semánticos (*llamados así porque recogen la semántica del dominio del problema*) y que son reconocidos en la fase de análisis.

Bajo esta perspectiva, prácticamente la construcción de cualquier tipo de aplicación software puede encuadrarse dentro de un proyecto software orientado a objetos.

Tradicionalmente, el soporte metodológico para la construcción de software orientado a objetos ha venido de la mano de alguno de los tres métodos más extendidos: **OMT** de **James Rumbaugh** [Rumbaugh et al., 1998], el **Método de Booch** de **Grady Booch** [Booch, 1996] u **OOSE (*Object-Oriented Software Engineering*)** de **Ivar Jacobson** [Jacobson et al., 1992].

Sin embargo, precisamente la unión de estos tres líderes de la orientación a objeto bajo el amparo de la empresa **Rational Software Corporation**, ha dado lugar al método unificado o **RUP** (*Rational Unified Process*) [Jacobson et al, 1999].

El proceso unificado presenta un marco de desarrollo en el que encuadrar cualquier tipo de aplicación software (*ya sea gestión, software en tiempo real, multimedia, basado en web…*). Este proceso se caracteriza por los siguientes rasgos:

- **Basado en casos de uso.**
- **Centrado en la arquitectura**
- **Iterativo**
- **Incremental**

Precisamente los casos de uso, aunque no exentos de problemas, pueden acomodarse muy bien para el desarrollo de aplicaciones dirigidas por eventos, como puede ser el caso de aplicaciones multimedia realizadas en entornos de desarrollo con un modelo objeto un tanto limitado.

3. Memoria del proyecto

La memoria de un proyecto software básicamente coincide con la documentación técnica elaborada en el proceso de construcción del software. En el caso que se está tratando, proyectos de final de carrera de una Ingeniería Técnica en Informática, la memoria estará formada por dos partes bien diferenciadas: una primera parte de descripción del proyecto realizado, que se asemeja a las memorias tradicionales presentadas para la consecución de los diferentes grados universitarios; y una segunda parte que recoja la documentación técnica del proyecto.

3.1 Parte 1 – Descripción del proyecto

La organización de esta parte dependerá mucho del tipo de proyecto y de la organización que los tutores del proyecto crean oportuna. En general, esta parte debe plantear el problema al que se ha dado solución, debe indicar los aspectos más interesantes que se han utilizado para solucionarlo y debe acabar con unas conclusiones del proyecto y las líneas de trabajo que puedan ampliar el proyecto realizado.

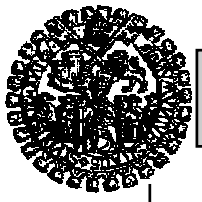
A continuación se presenta un esqueleto básico orientativo de lo que puede ser esta primera parte.

1. Introducción
2. Objetivos del proyecto
3. Conceptos teóricos
4. Técnicas y herramientas
5. Aspectos relevantes del desarrollo del proyecto
6. Trabajos relacionados
7. Conclusiones y líneas de trabajo futuras

Cuadro 1. Apartados de la primera parte de la memoria.

3.1.1 Apartado de introducción

En este primer apartado se hace una breve presentación del tema que se aborda en el proyecto, a la vez que se hace una breve descripción del contenido y estructura de la memoria.



3.1.2 Apartado de objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que se plantea el alumno a la hora de llevar a la práctica el proyecto.

3.1.3 Apartado de conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

3.1.4 Apartado de técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas.

No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

3.1.5 Apartado con los aspectos relevantes del desarrollo

Este apartado pretende recoger los aspectos más interesantes del desarrollo del sistema software centro del proyecto, comentados por los autores del mismo.

Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación.

Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales.

Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas[2], normalización y desnormalización, distribución en ficheros[3], reglas de negocio dentro de las bases de datos (*bases de datos activas*), aspectos de desarrollo relacionados con el WWW...

---

[2] Una referencia al tema de selección de índices es el capítulo 3 del [Shasha, 1992] (*además el capítulo 4 es muy aconsejable para el tema de la desnormalización*).

[3] Uno de las referencias más interesantes en este apartado son los capítulos 3 y 4 del [Loney, 1995]. Aunque es un libro orientado a ORACLE, sus contenidos se pueden aplicar a otros servidores de

Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

3.1.6 Apartado de trabajos relacionados

Este apartado sería parecido a lo que en trabajos de grado (*tesinas y tesis*) se denomina estado del arte. En un proyecto de final de carrera de primer ciclo no parece obligada su presencia (*a diferencia del segundo ciclo, donde la justificación teórica mediante un estado del arte debe acompañar a los productos software realizados*), aunque se puede dejar a juicio del tutor o tutores el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

3.1.7 Apartado de conclusiones y líneas de trabajo futuro

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas.

Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

3.2 Parte 2 – Documentación técnica

Los contenidos de esta parte van a estar totalmente condicionados por la metodología de desarrollo que se haya seguido, la cual habrá ido indicando que documentos se han debido de ir generando.

Cada uno de estos documentos debe incluirse como un anexo a la memoria del trabajo, de forma que el conjunto de anexos constituyan la documentación técnica del proyecto.

Anexo 1. Plan del Proyecto Software
Anexo 2. Especificación de Requisitos del Software
Anexo 3. Especificación de Diseño
Anexo 4. Documentación Técnica de Programación
Anexo 5. Manuales de Usuario

Cuadro 2. Documentación técnica.

bases de datos. Por supuesto, el apartado de distribución de ficheros carece de sentido en proyectos realizados sobre Sistemas Gestores de Bases de Datos de sobremesa (*tipo MS Access o Paradox*) o cuando la máquina de explotación tiene un único disco duro.

A modo de guía orientativa, y sin seguir ninguna metodología de desarrollo en concreto, en el Cuadro 2 se presenta un esquema de anexos que pueden conformar la documentación técnica de un proyecto tipo.

3.2.1 Anexo 1 – Plan del proyecto software

En los proyectos de fin de carrera de las Ingenierías Técnicas en Informática, no se suele hacer mucho hincapié en la parte que de forma genérica podía denominarse *gestión de proyectos*. No obstante, no es descabellado ir introduciendo al alumno de primer ciclo en la realización de estas actividades, buscando crear el hábito necesario en estas actividades de cara a un hipotético segundo ciclo o para su participación en proyectos reales. Toda la documentación generada en estas actividades pertenece a la documentación técnica del mismo.

De forma general, se puede encontrar el esquema de este documento en el capítulo 5 de [Piattini et al., 1996].

Centrando la atención en la realización práctica de la mayoría de los proyectos de fin de carrera, puede ser interesante que la documentación de este apartado se dirija hacia dos puntos concretos: *la planificación temporal del proyecto* y *el estudio de viabilidad del proyecto*.

- **Planificación temporal del proyecto:** Se recoge la planificación temporal del proyecto o lo que es lo mismo la elaboración del calendario o programa de tiempos.
  El principal objetivo de esta tarea es recoger de una forma gráfica todas las actividades del proyecto necesarias para producir el resultado final.

  El destinatario de este documento es el gestor del proyecto, de forma que le facilite su labor de dirección. En el caso de los proyectos de fin de carrera, dado su pequeño tamaño y su reducido número de participantes, se justifica más su presencia por el hecho de que los alumnos aprendan a planificar un calendario, y por la experiencia, en forma de métricas, que estos calendarios pueden aportar para estimaciones de futuros proyectos de mayor entidad.

  Se aconseja utilizar técnicas gráficas como **diagramas de Gantt** o **redes PERT** [Piattini et al., 1996], [Pressman, 1998]**.**

- **Estudio de viabilidad del proyecto:** No es infrecuente que en determinados proyectos de fin de carrera vinculados de alguna forma con empresas reales, se solicite un estudio de viabilidad, normalmente centrado en la viabilidad económica del mismo[4]. Precisamente, en relación con la viabilidad económica, la técnica que mayor información puede aportar es el *análisis coste/beneficio*, que ofrece una valoración de la justificación económica para un proyecto software.

[4] La viabilidad de un proyecto software se centra en cuatro áreas de interés: *la viabilidad económica, la viabilidad técnica, la viabilidad legal* y *el estudio de alternativas.*

En el Cuadro 3 se ofrece el esquema general, basado en el propuesto por Roger S. Pressman [Pressman, 1998], para un documento que recoja el estudio de viabilidad de un proyecto software; el cual puede servir como referencia genérica a adaptar en cada caso.

Portada Lista de cambios Tabla de contenidos Lista de figuras Lista de tablas Introducción Resumen de gestión y recomendaciones Alternativas Descripción del sistema Análisis coste/beneficio Evaluación de riesgo técnico Consideraciones legales Otros temas específicos del proyecto Glosario

Cuadro 3. Esquema general del documento de estudio de viabilidad.

3.2.2 Anexo 2 – Especificación de requisitos del software

Este documento cumple dos cometidos básicos, en primer lugar de cara al cliente[5] debe recoger todos los requisitos del sistema software a construir, de forma que sirva como elemento contractual entre la parte cliente y los ingenieros del software. En segundo lugar, debe servir para especificar estos requisitos con los métodos gráficos, textuales o formales que se crean más oportunos, presentando todas las vistas o enfoques del sistema (*funcional, de información, de comportamiento*) como se precisen para completar el análisis.

Las distintas aproximaciones metodológicas ofrecen distintas variantes a este documento, presentando en ocasiones dos documentos diferenciados, uno con el catálogo de requisitos, que suelen denominar *Documento de Requisitos del Sistema* y otro documento con las especificaciones detalladas de estos requisitos.

Existen diversos formatos para la creación de este documento, algunos de los cuales aceptados internacionalmente, como es el caso del **IEEE Std. 830-1998** [IEEE, 1999]. De nuevo, en el Cuadro 4 se presenta una propuesta de formato para el documento de especificación de requisitos, sin seguir ninguna metodología, buscando que sirva como guía orientativa genérica, independientemente del paradigma elegido.

[5] Aquí el término cliente se refiere en abstracto al interesado que ha causado que el proyecto se lleve a cabo (*tutor, empresa, departamento...*). Es decir, equivalente a lo que en la bibliografía inglesa se denomina *stakeholder.*

- **Portada:** que recoja el tipo de documento, el nombre del proyecto, la versión, la fecha de la versión, los responsables de la realización del documento y el nombre del cliente (*en caso de existir un destinatario del proyecto*).

Portada
Lista de cambios
Tabla de contenidos
Lista de figuras
Lista de tablas
    Introducción
    Objetivos del proyecto
    Lista de usuarios participantes
    Descripción del sistema actual
    Catálogo de requisitos del sistema
    Especificaciones de requisitos
        Especificaciones funcionales
        Especificaciones de información
        Especificaciones de eventos
        Especificaciones no funcionales
    Interfaces de usuario
    Pruebas
Glosario
Apéndices

Cuadro 4. Estructura del documento de especificación de requisitos del software.

- **Lista de cambios[6]:** donde se especifiquen, para cada versión del documento, las causas de los cambios producidos en el mismo. Para cada cambio se debe incluir el número del cambio, la fecha del cambio, la descripción del cambio y el autor o autores del mismo, tal y como se muestra en la Tabla 1.
- **Tabla de contenidos:** indicando la página en que comienza cada apartado del documento, con las sangrías oportunas de cara a facilitar la comprensión de la estructura del documento.
- **Lista de figuras y lista de tablas:** si el número de figuras y/o de tablas es importante, deberán incluirse las listas de figuras y/o de tablas oportunas.

[6] Es una buena costumbre que todo documento técnico cuente con una lista de cambios, porque refleja la evolución que ha ido sufriendo en proyecto reflejado en los cambios que se recogen en el documento. Esto es una constatación más de que la documentación técnica debe irse desarrollando a la par que el proyecto en curso (*pudiendo sufrir alteraciones en su ciclo de vida*), y no al final del mismo como un compromiso hacia unos determinados estándares o imposiciones.

Núm.	Fecha	Descripción	Autor/es
0	$fecha_0$	Versión 1.0	$autor_0$
1	$fecha_1$	$descripción\ cambio_1$	$autor_1$
⋮	⋮	⋮	⋮
n	$fecha_n$	$descripción\ cambio_n$	$autor_n$

Tabla 1. Lista de cambios de la Especificación de Requisitos del Software.

- **Introducción:** apartado que realiza una breve presentación del trabajo, describiendo brevemente el contenido y la estructura del documento.
- **Lista de usuarios participantes:** recogiendo el conjunto de personas con las que se ha tenido contacto para la realización del análisis de requisitos, así como su relación con el sistema a construir.
- **Descripción del sistema actual:** apartado que recoge toda la información pertinente al sistema actual, en caso de existir un sistema ya implantado. Pueden recogerse aquí modelos físicos y lógicos (*tanto de procesos como de información*) del sistema existente, así como una lista de problemas y necesidades.
- **Catálogo de requisitos del sistema:** de forma textual se recogen la lista de requisitos funcionales, de información y de otros tipos, acompañando a cada uno de ellos con una breve descripción textual y algún indicador de su prioridad e importancia.
- **Especificación de los requisitos:** esta parte recoge todo el conjunto de modelos y técnicas de especificación utilizadas para especificar los requisitos anteriormente descritos. Es la parte que más va a variar en función del método de análisis empleado y de la tipología del proyecto. En general, se puede hablar de especificaciones de requisitos funcionales, especificaciones de requisitos de información, especificaciones de eventos y especificaciones de requisitos no funcionales.
- **Interfaces de usuario:** conjunto de ventanas e informes que constituyen la primera aproximación a la interfaz de usuario.
- **Pruebas:** definición de las pruebas de aceptación del sistema.
- **Glosario:** apartado de carácter opcional que tiene el cometido de recoger una lista ordenada alfabéticamente de los acrónimos y términos que aparezcan en el documento cuyo significado debe ser aclarado.

- **Apéndices:** que aportan información adicional que se incluye separada de la documentación obligatoria del documento. Suelen tener el cometido de recoger documentos proporcionados por el cliente (*formularios, ejemplos de listados, fichas...*), lo cual lleva a que en muchas ocasiones se traten de fotocopias que se incluyen como complemento.

  Otro posible apéndice puede ser el que recoja las transcripciones o las actas de las entrevistas realizadas a los clientes.

  Sólo deben aparecer si se consideran oportunos y se identifican con letras ordenadas alfabéticamente.

3.2.3 Anexo 3 – Especificación de diseño

Este documento supone la entrada en el dominio de la solución del problema, es decir, comienza la fase tradicional del desarrollo del sistema software. El paso del análisis al diseño es algo que en muchas ocasiones es difícil de plasmar en los modelos, especialmente cuando se está trabajando en orientación a objetos, donde los modelos evolucionan plasmando un refinamiento de sus contenidos desde la fase de análisis a la fase de implementación[7].

En general, se puede decir que el documento de especificación de diseño marca el camino de solución a seguir, tomándose decisiones muy importantes dentro de la arquitectura, los datos, la interfaz o los detalles procedimentales de la aplicación software.

Los detalles de diseño son fundamentales para conseguir los factores de calidad externos e internos que marcarán la calidad del producto software final.

De nuevo, siguiendo la filosofía de este documento, en el Cuadro 5 se incluye un esquema de un posible documento de especificación de diseño tipo, intentando ser lo más independiente posible de cualquier metodología y/o paradigma de desarrollo, de forma que se pueda adaptar la estructura propuesta a la casuística exigida por cada proyecto particular.

La estructura presentada en el Cuadro 5 puede calificarse de muy influenciada por el paradigma estructurado, aunque no es difícil obtener una estructura equivalente para el paradigma orientado a objetos incorporando, por ejemplo, el concepto de arquitectura de 4+1 vistas [Kruchten, 1995][8].

---

[7] Como consideración práctica a seguir cuando se esté trabajando con orientación a objeto, se puede establecer que aquellos modelos que se centren en recoger los objetos semánticos del dominio del problema pertenecerán a la fase de análisis, mientras que aquéllos que refinen dichos objetos con matices más propios del dominio de la solución serán considerados como modelos de diseño.

[8] Este modelo propone utilizar cinco vistas concurrentes para describir la arquitectura software:

- Vista lógica: Describe el modelo de objetos de diseño.
- Vista de procesos: Describe el diseño de los aspectos de concurrencia y de sincronización.

Portada
Lista de cambios
Tabla de contenidos
Lista de figuras
Lista de tablas
- Introducción
- Ámbito del software
- Diseño de datos
- Diseño arquitectónico
- Diseño de la interfaz
- Diseño procedimental
- Referencia cruzada a los requisitos
- Pruebas
- Entorno tecnológico del sistema
- Plan de desarrollo e implementación

Glosario
Apéndices

Cuadro 5. Estructura del documento de especificación de diseño.

Cuando se desarrolle una aplicación bajo el paradigma objetual, otra opción a la hora de documentar el diseño de la misma es la utilización de patrones de diseño [Gamma et al., 1995].

- **Portada:** que recoja el tipo de documento, el nombre del proyecto, la versión, la fecha de la versión, los responsables del documento y el nombre del cliente (*en caso de existir un destinatario del proyecto*).
- **Lista de cambios:** donde se especifiquen para cada versión del documento, las causas de los cambios producidos en el mismo. Para cada cambio se debe incluir el número del cambio, la fecha del cambio, la descripción del cambio y el autor o autores del mismo, tal y como se muestra en la Tabla 1.
- **Tabla de contenidos:** indicando la página en que comienza cada apartado del documento, con las sangrías oportunas de cara a facilitar la comprensión de la estructura del documento.
- **Lista de figuras y lista de tablas:** si el número de figuras y/o de tablas es importante, deberán incluirse las listas de figuras y/o de tablas oportunas.

- Vista física o de despliegue: Describe la correspondencia entre el software y el hardware, reflejando los aspectos de distribución.
- Vista de realización: Describe la organización estática del software en su entorno de desarrollo.
- Vista de los casos de uso: Unifica las cuatro vistas anteriores.

- **Introducción:** apartado que realiza una breve presentación del trabajo, describiendo brevemente el contenido y la estructura del documento.
- **Ámbito del software:** el objetivo de este apartado es recoger someramente de una forma textual los objetivos del sistema software, los principales requisitos y las restricciones de diseño.
- **Diseño de datos:** el cometido del diseño de datos es seleccionar representaciones lógicas de los objetos de datos (*estructuras de datos*) identificados en la fase de análisis de requisitos. Dependiendo del tipo de aplicación que se esté desarrollando, se pueden tener diferentes aproximaciones en cuanto a los datos, pudiéndose distinguir en general:
  - ***Objetos de datos y estructuras de datos:*** manteniendo un diccionario de datos que represente las relaciones entre los objetos de datos y las restricciones de los elementos de una estructura de datos.
  - ***Estructuras de archivo y bases de datos:*** actualmente, en la inmensa mayoría de aplicaciones que manejan bases de datos se trabaja con el modelo relacional, presentando el esquema lógico de los datos de sistema (*en tercera forma normal o en la forma normal de Boyce-Codd*), la estructura de tablas y de vistas, los ficheros auxiliares y la descripción de los atributos.

    Dentro del diseño de datos también puede darse el caso de realizar actividades propias del diseño físico[9] de los datos, esto es, actividades relacionadas con el diseño físico de los datos: desnormalización, creación de índices, *clustering* de tablas...
- **Diseño arquitectónico:** el objetivo del diseño arquitectónico es desarrollar la estructura modular, representando las relaciones de control de los módulos, combinando la estructura de programa con la estructura de datos, definiendo las interfaces que permiten el flujo de datos a través del programa. Cuando se trabaja en el paradigma estructurado resulta interesante incorporar la jerarquía de módulos con sus interfaces. Cuando se trabaja en orientación a objeto, la descripción de la arquitectura puede hacerse describiendo los paquetes y sus relaciones, para posteriormente describir cada una de las clases de los paquetes y sus relaciones.
- **Diseño de la interfaz:** la interfaz hombre-máquina, que puede haber sido inicialmente esbozada en el análisis de requisitos en diferentes prototipos, como apoyo a la obtención del conocimiento sobre el dominio de la aplicación, debe

[9] Se recuerda que el proceso de creación de una base de datos tiene tres pasos diferenciados: el modelado conceptual (*utilizando un diagrama entidad/relación o un diagrama de clases de UML por ejemplo*), el modelado lógico (*tablas, normalización, diagramas de Bachmann en Codasyl...*) y el modelado físico (*desnormalización, creación de índices*). Mientras que el modelado conceptual se vincula a la fase de análisis, el modelado lógico y físico se vincula con la fase de diseño. Una excelente referencia en este sentido la constituyen los capítulos 22, 23 y 24 del [DeMiguel y Piattini, 1993].

quedar totalmente cerrada en el diseño, presentando, si se cree oportuno, los aspectos más relevantes de ésta en el documento de especificación de diseño.

La proliferación de los entornos gráficos hace que la mayoría de las aplicaciones se basen en el manejo de formularios con un alto número de controles que se activan y desactivan en función de la operación que se esté realizando en cada momento. Una forma bastante adecuada para la descripción y documentación de las posibles combinaciones activación/desactivación que ofrecen los controles de un formulario es establecer una tabla NxN, siendo N el número de controles, estableciendo las relaciones entre ellos [González, 1999].

- **Diseño procedimental:** debe recoger los detalles de los algoritmos mediante algún método que evite ambigüedades. En la mayoría de los casos se puede utilizar para describir los algoritmos más importantes del sistema software, no siendo desaconsejable para esta tarea la utilización de construcciones derivadas del lenguaje de programación que finalmente se vaya a utilizar. Realmente, en la mayoría de estas ocasiones este diseño detallado se realiza directamente sobre la herramienta de desarrollo en la que se va a implementar el producto software, quedando reflejado este diseño detallado en el código fuente debidamente comentado.
- **Referencia cruzada con los requisitos:** es muy interesante de cara a la trazabilidad incluir algún tipo de técnica matricial o gráfica que relacione los requisitos funcionales con los elementos de diseño (*módulos, paquetes…*). Con estas referencias se consigue estudiar si todos los requisitos han sido satisfechos en la etapa de diseño y señalar qué módulos son críticos para implementar requisitos específicos.
- **Pruebas:** apartado que recoge las directrices para probar los módulos individuales y su integración en subsistemas, así como las pruebas de integración de los subsistemas en el sistema completo.
- **Entorno tecnológico del sistema:** estableciendo por una parte el equipo físico, el equipo lógico y las comunicaciones que marcan el contexto del sistema software, y por otra todas las restricciones técnicas que existan.
- **Plan de desarrollo e implantación:** se establecen las directrices a seguir en la implementación del sistema, así como su implantación en el entorno de explotación. Dentro de este plan, y muy estrechamente relacionado con el diseño físico de los datos, está (*si fuera necesario*) la distribución de los datos a través de diferentes ficheros y discos (*incluso máquinas y lugares geográficamente distantes*).
- **Glosario:** apartado de carácter opcional que tiene el cometido de recoger una lista ordenada alfabéticamente de los acrónimos y términos que aparezcan en el documento cuyo significado debe ser aclarado.

- **Apéndices:** que aportan información adicional que se incluye separada de la documentación obligatoria del documento. Sólo deben aparecer si se consideran oportunos y se identifican con letras ordenadas alfabéticamente.

3.2.4 Anexo 4 - Documentación técnica de programación

Este documento se suele asociar al código fuente de la aplicación, aunque puede llegar a ser más amplio dependiendo del proyecto, siendo posible encontrar los apartados presentes en el Cuadro 6.

Portada
Lista de cambios
Tabla de contenidos
Introducción
Documentación de las bibliotecas
Código fuente
Manual del programador
Pruebas unitarias
Glosario
Apéndices

Cuadro 6. Documento técnico de programación.

- **Portada:** que recoja el tipo de documento, el nombre del proyecto, la versión, la fecha de la versión, los responsables del documento y el nombre del cliente (*en caso de existir un destinatario del proyecto*).
- **Lista de cambios:** donde se especifiquen para cada versión del documento, las causas de los cambios producidos en el mismo. Para cada cambio se debe incluir el número del cambio, la fecha del cambio, la descripción del cambio y el autor o autores del mismo, tal y como se muestra en la Tabla 1.
- **Tabla de contenidos:** indicando la página en que comienza cada apartado del documento, con las sangrías oportunas de cara a facilitar la comprensión de la estructura del documento.
- **Introducción:** apartado que realiza una breve presentación del trabajo y de la plataforma de desarrollo utilizada, describiendo también brevemente el contenido y la estructura del documento.
- **Documentación de las bibliotecas:** si para la realización del producto software (*o el propio el contenido del producto final es una o varias bibliotecas*) se han realizado diferentes bibliotecas (*de funciones de utilidad, de clases…*) éstas deben ser documentadas en formato de guía de referencia.
- **Código fuente:** es la parte fundamental de este documento, aunque no se incluye como parte de la memoria, sino incluido en soporte magnético como complemento a la memoria. Puede ser interesante incluir un apartado en el que se

presente la estructura de ficheros que forman el código fuente, con una pequeña descripción de los contenidos de cada fichero.

El código fuente debe considerarse como un conjunto de documentos que forman parte de la documentación técnica del proyecto, por lo que se debe cuidar el formato del mismo, de forma que éste cumpla las características siguientes:

- *Legible*: Buscar un código que sea fácil de leer (*utilizando un sangrado adecuado, incluyendo líneas en blanco, siendo constantes en la utilización de los nombres de los identificadores, no abusando de líneas demasiado largas, utilizando comentarios…*).
- *Autocomentado*: La utilización de comentarios no superfluos es una característica importante que ayuda a la legibilidad del código, pero igual de importante es la utilización de nombres significativos para identificadores de variables, constantes y funciones.

Igualmente importante para la legibilidad del código es la utilización de una documentación interna, que complemente a las características de código autocomentado, y que se consigue utilizando cabeceras para cada fichero/módulo y para cada función.

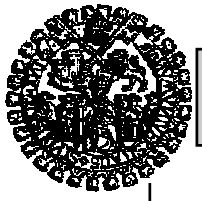
Para cada fichero/módulo se sugiere un marco formado por las siguientes entradas:

Nombre
Sinopsis
Directivas de compilación
Lista de códigos de salida de la aplicación
Dependencias funcionales
Lista de situaciones críticas
Organización[10]
Autor/autores
Fecha
Versión
Lista de versiones
Referencia a las fuentes consultadas

Para cada función se sugiere un marco formado por las siguientes entradas:

Nombre
Sinopsis
Ejemplo de llamada
Argumentos de entrada
Valores de retorno
Dependencias funcionales
Pseudocódigo
Autor
Fecha
Lista de versiones
Referencia a las fuentes consultadas

[10] En el caso de los proyectos de final de carrera se utilizará este campo para indicar que se trata de un código fuente vinculado a un proyecto de final de carrera realizado en el seno de un determinado departamento de una Universidad. En casos de colaboración Universidad/Empresa, además de las indicaciones anteriores aparecerá la empresa interesada.



Es muy interesante seguir alguna de las guías de estilo propias de cada lenguaje de programación.

- **Manual del programador:** recogiendo todas las acciones necesarias para generar el producto final a partir del código fuente (*opciones de compilación, makefiles…*).
- **Pruebas unitarias:** descripción de las pruebas unitarias realizadas, insertándolas en el documento o comentándolas brevemente e incluyéndolas en el soporte magnético junto al código fuente.
- **Glosario:** apartado de carácter opcional que tiene el cometido de recoger una lista ordenada alfabéticamente de los acrónimos y términos que aparezcan en el documento cuyo significado debe ser aclarado.
- **Apéndices:** que aportan información adicional que se incluye separada de la documentación obligatoria del documento. Sólo deben aparecer si se consideran oportunos y se identifican con letras ordenadas alfabéticamente.

3.2.5 Anexo 5 - Manuales de usuario

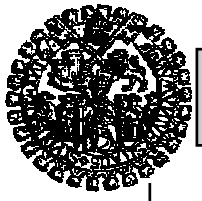
En este anexo se recogen todos los tutoriales, manuales y guías de usuario que sean necesarios para el correcto manejo de la aplicación. Se recomienda utilizar los formatos adecuados para su fácil transformación en ficheros de ayuda en línea de la aplicación, así como para su difusión y consulta en los formatos más difundidos (*HTML, PostScript, PDF, RTF…*).

Aunque el número y el contenido de los documentos de este anexo puede ser muy diverso y variado dependiendo del proyecto, en la mayoría de los casos siempre deberían aparecer los siguientes:

- **Documento de instalación y configuración:** que indique cómo instalar el programa en el ordenador de explotación. Debe incluir una especificación precisa y clara de los requisitos hardware necesarios (*así como una lista de incompatibilidades si existiera*), referencia al sistema operativo sobre el que se va a ejecutar (*con indicación precisa de versión, protocolos de red, bibliotecas necesarias...*), tipo y versión de navegador, en caso de requerirlo, (*diferenciando que opciones de configuración debe ser establecidas en cada caso*)...

  También podrían incluirse todos los detalles relativos a la distribución de los ficheros de la base de datos, facilitando y documentando los *scripts* de generación de índices, carga de datos en la base de datos...

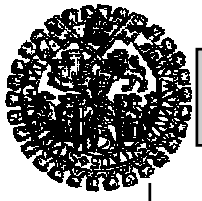
  En relación con este apartado es de agradecer la realización de una herramienta de instalación interactiva que facilite el proceso de instalación.
- **Manual de usuario:** que incluya todas las opciones del programa.



3.3 Otros apartados de la memoria

La memoria, a parte de la descripción del proyecto y de la documentación técnica, puede contener otra serie de apartados como documento que es. Entre los de especial interés cabe destacar:

- **Portada:** que recoja el escudo de la Universidad, el nombre del centro donde está adscrita la titulación, el nombre de la titulación y la Universidad donde se cursa, el grado al que se opta con el trabajo, el título del mismo, el nombre del autor o autores, el nombre del tutor o tutores, el departamento bajo el cual se ha realizado y la fecha. Si el proyecto ha sido realizado en alguna empresa, debe aparecer una reseña a la empresa donde ha sido (*o para la que ha sido*) desarrollado, además del nombre del tutor(es) en la empresa. En este documento de forma intencionada no se pretenden establecer normas de estilo, siendo en el seno de cada Escuela o Facultad donde se dictarán los estándares a seguir, buscando una uniformidad de estilo y presentación de los proyectos de fin de carrera.
- **Certificado del tutor/tutores:** hoja con la certificación de los tutores de la realización del proyecto.
- **Agradecimientos:** frases de agradecimiento.
- **Tabla de contenidos:** indicando la página en que comienza cada apartado del documento, con las sangrías oportunas de cara a facilitar la comprensión de la estructura del documento.
- **Lista de figuras y lista de tablas:** si el número de figuras y/o de tablas es importante, deberán incluirse las listas de figuras y/o de tablas oportunas.
- **Lista de acrónimos:** listado alfabético de los acrónimos que aparecen en la memoria.
- **Glosario:** apartado de carácter opcional que tiene el cometido de recoger una lista ordenada alfabéticamente de los términos que aparezcan en el documento cuyo significado debe ser aclarado.
- **Apéndices:** que aportan información adicional que se incluye separada de la documentación obligatoria del documento. Sólo deben aparecer si se consideran oportunos y se identifican con letras ordenadas alfabéticamente. Puede ser interesante incluir apéndices que describan herramientas de apoyo utilizadas, APIs y frameworks empleados, herramientas de utilidad que se hayan construido como apoyo al proyecto y una descripción del contenido del soporte de almacenamiento que acompaña a la memoria (*típicamente un CD-ROM*).
- **Bibliografía:** apartado que debe recoger todas las fuentes bibliográficas. Tradicionalmente se incluyen todas aquellas que han sido referenciadas a lo largo de la memoria, aunque dado el carácter del documento pueden incluirse aquellas fuentes consultadas que no han sido citadas (*típicamente manuales y guías de*



programación y consulta). Debe consultarse con el tutor el formato de las citas bibliográficas, siendo los más comunes el estilo ACM o el Apalike. Además, dada la proliferación documentos electrónicos, es cada vez más frecuente tener que citar este tipo de referencias, en este sentido se recomienda seguir las normas para citar documentos electrónicos recogidas en [Estivill y Urbano, 1997]. **Es muy importante cuidar la bibliografía del proyecto**.

- **Índice alfabético:** índice inverso con los términos más importantes del proyecto.

3.4 Soporte para el almacenamiento de los ficheros del proyecto

Es necesario acompañar la memoria del proyecto con un soporte físico de almacenamiento que contenga el código fuente y los binarios que forman el producto software final, recomendándose para ello el uso de CD-ROMs.

Para dar un valor añadido al proyecto, y aprovechando la capacidad de almacenamiento que aporta el CD-ROM, éste puede incluir:

- **Código fuente:** del producto software.
- **Binarios:** del producto software. Si se ha desarrollado algún tipo de herramienta de instalación es una buena idea dejarlo preparado para que se pueda proceder a instalarlo desde el propio CD-ROM. Si es posible su ejecución desde el CD-ROM, indicarlo y prepararlo para que así sea.
- **Memoria del proyecto:** en el formato del procesador de textos utilizado, aunque sería interesante incluirla en otros formatos que permitan su fácil distribución si fuera oportuno (*SGML, HTML, PDF, PostScript...*).
- **Manuales de usuario:** en un formato fácilmente accesible desde el CD-ROM (*HTML o PDF*).
- **Entorno de desarrollo:** si es de libre distribución, incluir el entorno de desarrollo con que se ha desarrollado la aplicación.
- **Herramientas de utilidad:** si se han construido herramientas de apoyo al proyecto incluirlas con su código fuente y su formato binario.
- **Documentación de apoyo:** artículos, guías, manuales... relacionados con el proyecto, disponibles en un formato electrónico.

4. Normas de Formato y Estilo

El formato del cuerpo y secciones del documento no son objeto de este documento, aunque es recomendable que exista un estándar de formato definido en cada titulación, que se encuentre desarrollador en un documento y disponible en una plantilla de Microsoft Word y/o en un estilo LaTex.

En la redacción del documento se seguirán las normas de estilo apropiadas para un documento técnico [Levine et al., 1991]. Como consideraciones generales se aconseja ceñirse, en lo posible, a los siguientes criterios:

- Utilización de títulos directos y completos.
- Empleo de párrafos cortos.
- Uso de frases directas y completas, minimizando en la medida de lo posible las oraciones intercaladas.
- En general debe hacerse acopio de un estilo de redacción objetivo e impersonal.

5. Agradecimientos

Este documento es una iniciativa del Departamento de Informática y Automática de la Universidad de Salamanca, a la que posteriormente se han unido el Área de Lenguajes y Sistemas Informáticos (*Departamento de Ingeniería Civil*) de la Universidad de Burgos y el Departamento de Informática de la Universidad de Castilla-La Mancha en la persona del Dr. Mario Piattini.

Desde aquí los autores quieren agradecer a todos aquellos profesores y alumnos que han adoptado esta guía para la dirección/realización de sus proyectos de final de carrera, así como sus siempre interesantes comentarios para hacer de esta guía un documento vivo y permanentemente actualizado.

Apéndice A – Esquema de la memoria del proyecto

Portada

Certificado del tutor/tutores

Tabla de contenidos

Lista de tablas

Lista de figuras

1. Introducción
2. Objetivos del proyecto
3. Conceptos teóricos
4. Técnicas y herramientas
5. Aspectos relevantes del desarrollo del proyecto
6. Trabajos relacionados
7. Conclusiones y líneas de trabajo futuras

Anexo 1. Plan del Proyecto Software

Anexo 2. Especificación de Requisitos del Software

Anexo 3. Especificación de Diseño

Anexo 4. Documentación Técnica de Programación

Anexo 5. Manuales de Usuario

Lista de acrónimos

Glosario

Apéndices

Bibliografía

Índice alfabético

Cuadro 7. Esquema de la memoria del proyecto.

Bibliografía

- **[Booch, 1996] Booch, Grady.** "*Análisis y Diseño Orientado a Objetos con Aplicaciones*". 2ª Ed. Addison-Wesley/Díaz de Santos, 1996.
- **[DeMiguel y Piattini, 1993] De Miguel, Adoración y Piattini, Mario G.** "*Concepción y Diseño de Bases de Datos. Del Modelo E/R al ModeloRelacional*". Ra-ma, 1993.
- **[Estivill y Urbano, 1997] Estivill, Assumpció y Urbano, Cristóbal.** "*Cómo Citar Recursos Electrónicos*". Information World en Español. (También disponible en http://www.ub.es/biblio/citae-e.htm). Septiembre, 1997.
- **[Gamma et al., 1995] Gamma, Erich, Helm, Richard, Johnson, Ralph y Vlissides, John**. "*Design Patterns. Elements of Reusable Object-Oriented Software*". Addison-Wesley, 1995.
- **[González, 1999] Gonzalez, Alfons.** "*Visual Basic: Programacion Cliente/Servidor*". 2º Edición. RA-MA, 1999.
- **[IEEE, 1999] IEEE.** "*IEEE Software Engineering Standards Collection 1999 Edition. Volume 4: Resourse and Technique Standards*". IEEE Computer Society Press, 1999.
- **[Jacobson et al., 1992] Jacobson, Ivar, Christerson, M., Jonsson, P. y Overgaad G.** "*Object Oriented Software Engineering*". Addison-Wesley, 1992.
- **[Jacobson et al., 1999] Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** "*The Unified Software Development Process*". Addison Wesley, 1999.
- **[Kruchten, 1995] Kruchten, P.** "*The 4+1 View Model of Architecture*". IEEE Software, 12(6):42-50. November, 1995.
- **[Levine et al., 1991] Levine, Linda, Pesante, Linda H., and Dunkie, Susan B.** "*Technical Writing for Software Engineering*". Curriculum Module SEI-CM-23. Software Engineering Institute – Carnegie Mellon University. November, 1991.
- **[Loney, 1995] Loney, Kevin.** "*ORACLE. Manual del Administrador*". Oracle-Press – McGraw Hill, 1995.
- **[MAP, 1995] Ministerio para las Administraciones Públicas.** "*Metodología Métrica v2.1. Guía de Referencia*". Tecnos, 1995.
- **[Piattini et al., 1996] Piattini Velthuis, Mario G., Calvo-Manzano, José A., Cervera, Joaquín y Fernández, Luis.** "*Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión*". Ra-ma, 1996.
- **[Pressman, 1998] Pressman, Roger S.** "*Ingeniería del Software: Un Enfoque Práctico*". 4ª Edición. McGraw-Hill, 1998.
- **[Rumbaugh et al., 1998] Rumbaugh, James, Blaha, Michael, Premerlani, William, Eddy, Frederick y Lorensen, William**. "*Modelado y Diseño Orientados a Objetos. Metodología OMT*". 2ª Reimpresión. Prentice Hall, 1998.
- **[Shasha, 1992] Shasha, Dennis E.** "*Database Tunning. A Principled Approach*". Prentice Hall, 1992.
- **[Yourdon, 1993] Yourdon, Edward.** "*Análisis Estructurado Moderno*". Prentice-Hall Hispanoamericana, 1993.
- **[Yourdon Inc., 1993] Yourdon Inc.** "*Yourdon™ Systems Method. Model-Driven Systems Development*". Prentice Hall International Editions, 1993.