La Ingeniería de Software Libre y sus Herramientas Aplicadas a Proyectos Informáticos

Mauro Callejas Cuervo

Facultad de Ingeniería, Universidad Pedagógica y Tecnológica de Colombia
Avenida Central del Norte Km. 2, Tunja, Boyacá, COLOMBIA.
maurocallejas@yahoo.com

Resumen: En este artículo se pone a consideración algunas herramientas utilizadas en la ingeniería de software libre para el desarrollo de proyectos informáticos a nivel académico y a nivel empresarial, es así como se plantea la conformación de *kerneles* de trabajo discriminados en las diferentes etapas que componen el ciclo de vida de desarrollo de *software*; presenta, además, las funciones y principales artefactos que se deben obtener al aplicar este innovador paradigma (ingeniería de *software* libre, ISL). Por último se comenta de manera breve algunos casos prácticos sobre aplicaciones implantadas en entidades de carácter nacional, haciendo uso de herramientas de software libre.

Palabras claves: Ingeniería de *software* libre, *kernel*, Herramientas de *software* libres, Metodología.

Abstract: This article presents some tools used in the free software engineering to develop computation projects at academic and enterprise level, meantime the conformation of work kernelless is discussed and discriminated according to the different stages of life cycle of the software development. This project shows additionally the functions as well as the principal gadgets which should be obtained when the innovating paradigm (Free Software Engineering, FSE) is applied. Finally, some practical cases are discussed concerning their application to national entities, using the tools of free software.

Key words: Free software engineering, kernel, tools of free software, methodology.

1. Introducción

Este documento muestra un nuevo enfoque, que permite observar de manera diferente la forma de desarrollar *software*, y, por ende, traza algunas pautas para que este trabajo de carácter colectivo se realice de manera eficiente. Así se pretende mostrar un juego lingüístico, como el concepto de *kernel* o núcleo de trabajo, sitio web para manejar el proyecto (la aplicación) y algunos otros términos, que se describen más adelante.

La distribución del escrito está dada de forma tal que el lector tiene una guía práctica y secuencial para adentrarse en conceptos básicos de las herramientas utilizadas en la ISL. En primera instancia se presentan aspectos generales del trabajo enfocado bajo ciertas normas de ingeniería que se deben seguir; luego se plantean algunas funciones de los grupos que hacen parte del equipo de trabajo; posteriormente se comentan algunos casos prácticos en los cuales se han utilizados técnicas y métodos de ingeniería de software libre. Y finalmente se dan algunas conclusiones y se plantea el trabajo futuro de investigación.

Algunos de los aportes aquí presentados son parte de la experiencia docente, en el desarrollo de asignaturas relacionadas con ingeniería de *software* y de proyectos de *software* libre dirigidos en trabajos de pregrado en ingeniería de sistemas [Callejas et al, 2005], así como en la presentación de ponencias en eventos internacionales relacionados [Callejas et al, 2005], [Callejas, 2005].

2. Enfoques o métodos de desarrollo de ingeniería de software libre

La Ingeniería de *Software* Libre (ISL) permite que la metodología para el desarrollo de aplicaciones se lleve a cabo de manera amplia, ya sea utilizando un enfoque estructurado de análisis y diseño [Witten et al, 1996], [Yourdon, 1990], [Kendall & Kendall, 1998], un enfoque orientado por objetos [Meyer, 1998] o algún otro tipo de paradigma; además no limita a los analistas y diseñadores a utilizar una técnica de modelado y diagramación, como UML[Jacobson et al, 1999] o el modelado estructurado, ni ofrece recomendaciones que permitan evaluar el nivel de calidad de una organización, como lo promueve The Capability Maturity Model, CMM [Paulk et al, 1993]. Más bien se fundamenta en que se debe trabajar en equipo, con el fin de fomentar una mayor participación de elementos para el desarrollo óptimo de aplicaciones, sin

dejar de lado la utilización de técnicas y herramientas que aquí se mencionan. Además, se debe tener en cuenta el tiempo y los recursos asignados para cumplir con las tareas involucradas, evitando la pérdida de tiempo o abandono de los proyectos.

Con la ISL se pretende promover el uso de sistemas operativos, lenguajes de programación, bases de datos y demás herramientas de *software* de carácter libre para la creación de aplicaciones.

2.1. Juego lingüístico sobre ingeniería de *software* libre[1]

Un aspecto por tener en cuenta en el nuevo tópico de la ingeniería de *software* libre es el término *kernel*. ¿Por qué *kernel* y no simplemente grupo?
La visión de *kernel* está dada en que es un grupo el que lo conforma, pero puede tener aportaciones valiosas a su alrededor, y allí es donde se evidencia el trabajo colaborativo o en comunidad, haciendo que cualquier aporte hecho fuera del grupo pueda ser compilado en su interior, con el fin de enriquecer el producto final. Lo anterior puede llegar a ser comparado con un Sistema Operativo (SO), donde el *kernel* administra y controla tanto *software* como hardware de manera rústica (modo consola), pero se puede y se debe llegar a crear herramientas que permitan al usuario una mayor interacción con el computador. Por ejemplo, programar un escritorio para Linux (KDE) hace que la interfaz sea más amigable al usuario y además ofrece un aporte significativo para mejorar la aplicación final.

El ciclo de vida para el desarrollo del *software* puede tomar algunos aspectos relacionados con metodologías ágiles o metodologías tradicionales [Pressman, 2002], según la naturaleza del proyecto, pero haciendo un especial énfasis en que el desarrollo debe ser iterativo e incremental [Larman, 2002], tal y como se observa en la figura 1, en donde se presenta un proceso cíclico, que muestra como el *kernel* de planeación orienta a los demás *kerneles* en un perfecto engranaje y estos a su vez desarrollan sus propias actividades, dando su inicio a cada una de las iteraciones en el análisis y llegando a la implantación de cada uno de los módulos. Al finalizar cada iteración pasa nuevamente al análisis de otro de los módulos. Se observa además como la comunidad libre se encuentra fuera de la integración de los *kerneles*, ya que ésta es la que da sus aportes y punto de vista al grupo general de trabajo (aporta a cada uno de sus *kerneles* cuando sea el caso).

También el término 'comunidad libre' juega un papel importante en este tópico; se refiere al grupo de personas que participan en actividades relacionadas con el análisis, diseño, implementación e implantación de aplicaciones informáticas, haciendo uso de herramientas de *software* libre, y quienes de manera voluntaria apoyan cualquier labor que tenga alguno de los *kerneles*.

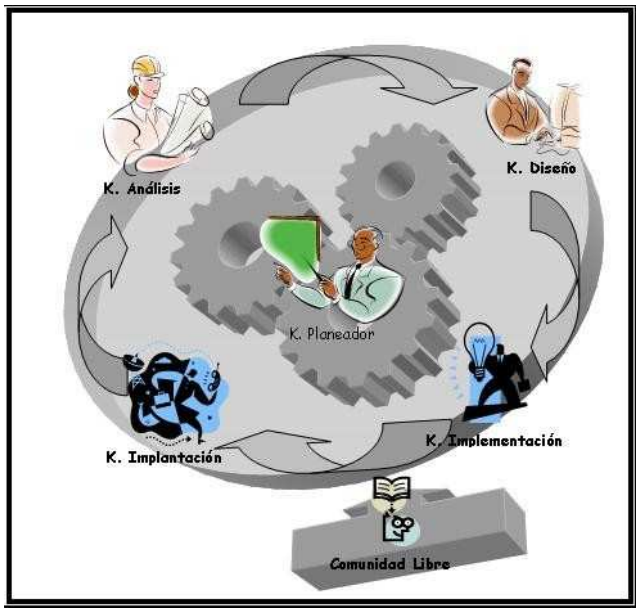

Fig. 1. Esquema de trabajo en Ingeniería de *Software* Libre.

Esta comunidad debe conocer las políticas y los lineamientos elaborados por el *kernel* planeador para su participación en el proceso de desarrollo de la aplicación.

2.2 Aspectos relevantes en el desarrollo de *software* libre

Cuando se generan aplicaciones de *software* libre es importante tener en cuenta las herramientas que se utilizarán para la divulgación y manipulación del desarrollo del proyecto (sitio web), así como las partes que componen el equipo de trabajo, sus funciones específicas y algunos otros aspectos que tienen que ver con la ingeniería de *software* tradicional. En este aparte se hace una breve descripción de cada una de ellas.

2.2.1 Crear un sitio WEB

Para generar esta herramienta de visualización y manipulación del proyecto se deben seguir ciertos principios de usabilidad, que se describen muy puntualmente en [Rodríguez de la Fuente et al, 2003]. En la web se plasmarán los avances, tareas pendientes, tareas asignadas, personas responsables y el cronograma de actividades fijado para el desarrollo total del proyecto; también debe mostrar la constitución de los diferentes grupos de trabajo colaborativo, con su líder correspondiente, quien será el responsable de la publicación de los diferentes hallazgos y productos resultantes del desarrollo de las actividades. El acceso de

[1] *Algunos de estos conceptos fueron presentados en el "IEEE 3er Congreso Internacional en Innovación y Desarrollo Tecnológico" realizado en el la ciudad de Cuernavaca, Morelos, México, en Septiembre de 2005, por Mauro Callejas Cuervo.*

cada grupo está controlado para las diferentes actualizaciones, inserciones, modificaciones y otras actividades en el proceso de creación del producto; el acceso para la comunidad libre participante será controlado y sus aportes serán recibidos en un espacio designado para tal fin.

2.2.2. Nombramiento de los diferentes grupos o *kernel*es de trabajo

Los *kerneles* que se integrarán tendrán funciones específicas y serán dirigidos por un líder nombrado por estos, quien hará las veces de vocero oficial de cada núcleo en las reuniones generales del proyecto. Los *kerneles* son:

***Kernel* de análisis**: nombrado *kernel* o núcleo porque está compuesto por una o varias personas que conforman la comunidad. Su líder guiará las diferentes actividades de análisis y será quien garantice la publicación de los productos generados en este grupo en el sitio web de la aplicación.

***Kernel* de diseño**: debe estar compuesto por un número limitado de miembros (dependiendo del volumen del proyecto); realizará las tareas de modelado lógico y físico de la aplicación y tendrá la responsabilidad de obtener los mejores modelos, poniéndolos a consideración de la comunidad libre, para captar sus aportes y recomendaciones.

***Kernel* de implementación**: se divide en varios sub*kernel*es, organizados según su distribución geográfica, con el objeto de obtener mayor cooperación interna dentro de cada uno de ellos y así poder enviar sus aportaciones al grupo planeador, coadyuvando a que los productos que salgan de un lugar específico (ciudad, departamento, país, continente) sean de alta calidad y puedan ser publicados en la web y puestos a disposición de los demás equipos de trabajo.

***Kernel* de implantación**: compuesto por un grupo concreto de personas identificadas totalmente, pues será el encargado de poner a consideración el producto y capacitar al usuario final.

Todos estos grupos o *kerneles* serán coordinados por un grupo denominado ***Kernel* de Planeación**, responsable de dar el visto bueno para el arranque y finalización de las actividades de la comunidad de cooperación libre y de certificar la liberación al mundo del producto final.

2.2.3. Asignación de funciones por *kernel*

A continuación se presentan algunas de las funciones más relevantes que se deben llevar a cabo en cada uno de los *kerneles*, así como también ciertos artefactos que se crearán al ejecutar dichas funciones.

***Kernel* de planeación:**

- Mantener el sitio web.
- Coordinar las tareas de inicio y finalización de actividades.
- Elaborar el cronograma de actividades para el ciclo de desarrollo del producto de *software* libre.
- Asignar los controles de acceso al sitio web de la aplicación a los diferentes grupos de trabajo.
- Coordinar tareas de empalme entre los diferentes grupos de trabajo.
- Mantener el directorio de cada uno de los miembros y sus roles dentro de los diferentes grupos de trabajo, así como la bitácora de colaboraciones y participaciones en el desarrollo de la aplicación.
- Definir cuál es el eje geográfico del dominio de la aplicación.
- Llevar el control de versiones del producto.
- Crear las políticas para actualizar el producto.
- Documentar los procesos en los que está relacionado.
- Liberar el producto para su uso.

***Kernel* de análisis:**

- Contextualizar el domino de la aplicación.
- Recolectar información relevante para el proyecto, haciendo uso de las diferentes técnicas de levantamiento de información (entrevistas, encuestas, documentación histórica).
- Generar el documento de requerimientos del sistema.
- Crear los modelos o diagramas preliminares del análisis (diagramas de flujos de datos, diagramas de funciones, diagramas de actividades o en orientación a objetos casos de uso).
- Recolectar los aportes colocados en el sitio web de la aplicación, en cuanto a análisis del problema, con el fin de procesarlos y determinar cuáles se tendrán en cuenta para su implementación.
- Generar la documentación que se desprende de cada actividad de análisis.

***Kernel* de diseño:**

- Generar los modelos lógicos y físicos.
- Implementar el diseño inicial de la interfaz gráfica de usuario.
- Recolectar los aportes colocados en el sitio web de la aplicación, en cuanto a diseño, con el fin de procesarlos y determinar cuáles se tendrán en cuenta para su aplicación.
- Generar la documentación que se desprende de cada actividad del diseño.

Es importante aclarar que en la ISL las especificaciones de diseño no se toman por decisión unilateral del *kernel* o por convicción propia de alguno de sus miembros, sino de forma cooperativa, es decir, los demás *kerneles* y los miembros de la comunidad darán su aprobación en conjunto con el usuario final.

***Kernel* de implementación:**

- Crear los subgrupos de codificación, asignando políticas claras de desarrollo, es decir, asignación de codificación por módulo, por formularios u otros métodos de división del trabajo de programación de la aplicación.
- Generar estándares de codificación y construcción de las interfaces, es decir, plantillas que busquen la unificación de criterios de programación; para esto se sugiere la implementación y uso de patrones de diseño [Gamma et al, 2002].
- Generar la codificación del producto.
- Desarrollar y llevar a cabo el plan de pruebas alfa.
- Generar la documentación que se desprende de la programación de los diferentes módulos de la aplicación. Esta documentación debe generarse internamente (dentro del código), como también de forma externa (en el manual del programador).

El trabajo que aquí se realiza es colaborativo, distribuido, y además debe haber un subgrupo que integre las diferentes partes de la aplicación ya programadas. Es de aclarar que un subgrupo puede estar trabajando en América y el otro en Europa o en cualquier parte del mundo.

***Kernel* de implantación:**

- Instalar la aplicación desarrollada.
- Generar y llevar a cabo el plan de pruebas beta del producto.
- Capacitar a la comunidad que utilizará la aplicación.
- Documentar los procesos que se llevaron a cabo en esta etapa.

Si se analiza con detenimiento, en este enfoque una de las principales funciones asignadas a cada grupo o *kernel* se fundamenta en la documentación exhaustiva de todos los procesos (análisis, diseño, implementación e implantación), pues es el valor agregado en una perspectiva de desarrollo colaborativo y de libertad para generar este tipo de aplicaciones.

3. Casos prácticos de desarrollo de aplicaciones informáticas e implementación del proceso de ISL

La propuesta nace del trabajo realizado para varias empresas e instituciones colombianas, en las cuales se desarrollaron sistemas de información haciendo uso de software libre para su implementación, y de la necesidad de mejorar los procesos de ingeniería de software tradicional, que se mostraron cortos en el momento de su aplicación.

De las entidades en las que se ha implementado software, tratando de llevar a cabo la aplicación de la ISL, se pueden nombrar: Caracol Radio S.A., Acerías Paz del Río, Instituto Nacional de Salud, varias instituciones educativas (software para colegios e institutos de educación no formal) y la Arquidiócesis de Tunja; comentaremos a continuación tres de los casos, desarrollados recientemente.

3.1 Caso Número Uno: Arquidiócesis de Tunja

En la Arquidiócesis de Tunja, Ciudad capital del Departamento de Boyacá, Colombia, se implementó, en el marco de la Asignatura Trabajo de Campo II, de cuarto año de Ingeniería de Sistemas de la Universidad Pedagógica y Tecnológica de Colombia (UPTC), la aplicación denominada "Sistema de control de información de la Arquidiócesis de Tunja", desarrollada con lenguaje PHP, base de datos MySql y utilizando como servidor de aplicaciones Apache, esta aplicación se encarga del manejo y control de la información de las diferentes parroquias que integran esta comunidad y de llevar el control de los diferentes eventos que allí se realizan. En el desarrollo de esta aplicación, los miembros del equipo se comunicaban a través del correo electrónico, adjuntando parte del código correspondiente al módulo asignado y poniendo a consideración del grupo su aportación; pero fue allí donde se notaron las falencias (limitación de espacio y tiempo), que fundamentaron la propuesta de crear equipos de trabajo (*kernel*es) y de condensar los avances en una página web que perteneciera al proyecto y en la cual se pudiera incluir los diferentes artefactos creados y se llevara un control riguroso del proceso, hasta obtener el producto terminado.

Los resultados obtenidos se reflejaron en la puesta en marcha de ésta aplicación. La carta de 'recibí a satisfacción del software', firmada por miembros de la Arquidiócesis de Tunja, que reposa en las oficinas de la UPTC, demuestra el trabajo óptimo que se realizó. Queda evidenciado que es necesario contar con un banco del proyecto en la web cuando los miembros involucrados en un equipo de desarrollo no se encuentran en el mismo sitio geográfico y que es importante la asignación de tareas claras para la consecución de los objetivos trazados.

3.2 Caso Número Dos: Caracol Radio S.A

El caso que se comenta en este numeral esta dado por el desarrollo de una práctica empresarial en Caracol Radio, la cual es una empresa colombiana líder en comunicación de la información, entretenimiento y servicio a través de la radiodifusión y nuevos medios electrónicos no convencionales, ésta práctica se tituló: "*El software libre como herramienta para el desarrollo de sistemas de información*", allí se investigó sobre herramientas de software libre para la generación de sistemas de información, además se realizó un aporte significativo al proceso de control de inventarios que se lleva a cabo en el Área de Servicios Generales de Caracol S.A. a través de la implementación de una herramienta informática guiada

por la metodología estructurada simplificada (MES), con una arquitectura cliente/servidor, se ofrece una solución a los requerimientos planteados por la Dirección de Compras, puesto que ha permitido conocer los activos fijos que tiene la Compañía a nivel nacional, de manera que exista mayor control sobre los recursos asignados a cada ciudad. El objetivo general fue: Analizar, diseñar e implementar un sistema de información utilizando herramientas libres (Linux, Kylix, Firebird), para el manejo y control de inventarios de las sedes de Caracol S.A. a nivel nacional.

El sistema desarrollado consta de una interfaz gráfica de usuario con un soporte de base de datos, que permite presentar una solución óptima al proceso de manejo y control de inventarios.

La aplicación se segmentó en tres funcionalidades: La primera corresponde a la gestión del sistema, donde únicamente el administrador de la base de datos tiene acceso, se encarga de insertar, eliminar, actualizar los usuarios del sistema; verificar las transacciones que se realiza en el sistema, permitiéndole consultar y auditar la información necesaria. La segunda corresponde al Almacén de Suministros, donde se encuentra el proceso de entrada y salida de elementos del almacén, se controla el stock mínimo, descripción detallada de cada artículo, estado en que se encuentra dentro del inventario. Y la tercera corresponde al Inventario, la cual provee al Analista y Jefe de compras toda la información necesaria para administrar los activos fijos y elementos almacenados en bodega.

La utilidad y confiabilidad al implementar software con herramientas libres se evidencia en el hecho que este *software* esta siendo utilizado actualmente.

3.3 Caso Número Tres: Instituto Nacional de Salud (INS)

El proyecto tratado en este caso se denominó "*Desarrollo e implementación del software en herramientas libres para RIPS*[2]*, según requerimientos del Instituto Nacional de Salud*", tuvo como fin ayudar a medir numéricamente la cantidad de enfermedades que se presentan en los municipios Colombianos, discriminadas por tipo y periodo en el cual se manifiestan. El software desarrollado es de libre uso para todos los municipios, sirve principalmente como apoyo a las investigaciones que desarrolla el INS sobre vigilancia y control de enfermedades de tipo hídrico; a la vez ayuda al fortalecimiento del conocimiento sobre el tratamiento y la calidad del agua potable en el control de enfermedades diarreicas.

El objetivo general de éste trabajo fue desarrollar e implementar un software en herramientas libres, que permitiera cargar y consultar los datos obtenidos de los archivos RIPS provenientes de las Entidades Prestadoras de Salud, tales como: hospitales, clínicas y diferentes centros médicos del país, que sirviera como apoyo para los estudios epidemiológicos que lleva a cabo el Instituto Nacional de Salud.

Para el desarrollo de la aplicación se utilizaron las siguientes herramientas de software libre:

- Lenguaje de programación: Java2 SDK, Standard Edition Versión 1.4.0.; IDE: NetBeans Versión 3.5.
- Base de datos: MySQL Versión 4.0, con WinMySQLAdmin, versión 1.4.
- Herramienta para diagramación: ArgoUML. 0.16.

A continuación se describe de forma breve la conformación de cada uno de los módulos que integran el software.

Módulo de Usuarios: Permite al administrador del sistema, crear, modificar y eliminar usuarios.

- Módulo de carga de archivos planos: A través de éste se ingresa la información de los archivos planos al banco de datos del aplicativo.
- Módulo medicamentos, diagnósticos y causas externas: Permite el ingreso de la información relacionada con los medicamentos, diagnósticos y causas externas.
- Módulo de Consultas: El módulo de consultas permite buscar la cantidad de ocurrencias de determinada enfermedad en un periodo de tiempo dado.
- Módulo de Reportes: Permite generar reportes de las consultas obtenidas por pantalla.

La mayoría de los centros prestadores del servicio de salud del País, no cuentan con suficientes recursos para el pago de licencias anuales para el funcionamiento de las aplicaciones implantadas, razón por la cual, los métodos, técnicas y herramientas del software libre se convierten en una solución práctica que demuestra una vez más las ventajas de esta actual tendencia.

4. Conclusiones

Actualmente el desarrollo de aplicaciones en comunidad no posee una clara definición de tareas, responsabilidades ni límites de trabajo documentadas, razón por la cual se llevó a cabo esta investigación basada en experimentos y casos prácticos, que permitieron mostrar algunas técnicas, métodos y herramientas para la implementación de aplicaciones informáticas, soportadas en el software libre.

[2] *RIPS: Registros Individuales de Prestación de Servicios en Salud*

El trabajo distribuido en *kernel*es es fundamental y se debe apoyar en la creación de un portal para condensar los resultados que se vayan obteniendo, dando como resultado una forma innovadora de plantear el desarrollo de *software*.

Los casos prácticos comentados en este artículo fueron el resultado del trabajo de investigación que se llevó a cabo en cumplimiento a los objetivos planteados por parte del Grupo de Investigación en Software - Proyecto *Software* Libre - UPTC, registrado en Colciencias.

5. Trabajos Futuros

Dentro del grupo de investigación se ha planteado la creación del proyecto "Municipio elite en software libre", cuyo objetivo es la sistematización (ofimática, desarrollo de aplicaciones) de cada una de las unidades administrativas en una localidad de cualquier municipio del País y por ende de instituciones gubernamentales similares. Dicho proyecto actualmente se encuentra en la fase de planeación.

6. Referencias

Callejas Cuervo, Mauro., Delgado Becerra, J. 2005. *El software libre como herramienta para el desarrollo de sistemas de información (Experiencia de una práctica empresarial en Caracol S.A.)*. Revista Ventana Informática, Edición Número 12, Ed. Universidad de Manizales.

Callejas Cuervo, M., et al., Ponencia *El software libre como herramienta para el desarrollo de sistemas de información*. IV Congreso Internacional de *Software* Libre GNU/LINUX, Universidad de Manizales, Colombia, Marzo de 2005.

Callejas Cuervo, M. Ponencia *Un tópico innovador en software: Ingeniería de software libre*. IEEE 3er. Congreso Internacional sobre Innovación y Desarrollo Tecnológico, Cuernavaca, Morelos, México, Septiembre de 2005.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. 2000. Patrones de Diseño, Ed. Addison-Wesley, 2002.

Jacobson, I., Booch, G., Rumbaugh, J. 1999. El Lenguaje Unificado de Modelado, Madrid, Ed. Addison Wesley.

Kendall & Kendall. Análisis y Diseño de Sistemas, Tercera Edición, Editorial Pearson.

Larman, C. 2002. Applying UML and Patterns. Ed. Prentice Hall.

Meyer, B. 1998. Construcción de *Software* Orientado a Objetos, Ed. Prentice-Hall.

Paulk, M. C., Weber, C. V., Curtis, B. y Chrissis, M. B. 1993. The Capability Maturity Model: Guidelines for Improving the *Software* Process, Ed. Addison-Wesley.

Pressman, R. 2002. Ingeniería del *Software*. Un enfoque práctico, Ed. MacGraw-Hill.

Rodríguez de la Fuente, Pérez, Carretero y otros. 2003. Programación de aplicaciones web, Ed. Thomson.

Witten, J., Bentely, L., Barlow, V. M. 1996. Análisis y diseño de sistemas de información. Ed. McGrawHill.

Yourdon, E. 1990. Análisis Estructurado Moderno, Ed. Campus.