DESARROLLO DE UN APLICATIVO WEB QUE PERMITA MONITOREAR LOS PROYECTOS Y PRODUCTOS DE INVESTIGACIÓN QUE REALIZAN LOS DOCENTES INVESTIGADORES DE LA FACULTAD DE INGENIERÍA

Presentado por:
VÍCTOR HUGO SOTO CALDERÓN

Presentado a:
DANIEL ARISTIZÁBAL TORRES
Dir. Investigaciones Facultad de Ingenierías

UNIVERSIDAD LIBRE SECCIONAL PEREIRA
FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
2013

Contenido

Índice de figuras

Índice de tablas

1. ANTECEDENTES

Un sistema de gestión de documentos es un sistema informático utilizado para rastrear y almacenar documentos electrónicos. Por lo general es también capaz de hacer el seguimiento de las diferentes versiones modificadas por los diferentes usuarios. El término tiene cierta superposición con los conceptos de los sistemas de gestión de contenidos. A menudo se considera como un componente de los sistemas de gestión de contenidos empresariales y en relación con la gestión de activos digitales, digitalización de documentos, sistemas de flujo de trabajo y los sistemas de gestión de registros.

A partir de finales de 1970, un número de vendedores comenzó a desarrollar sistemas de software para gestionar los documentos en papel. Estos sistemas tratan de documentos en papel, que incluye no sólo los documentos impresos y publicados, sino también fotografías, grabados, etc.

Más tarde, los desarrolladores comenzaron a escribir un segundo tipo de sistema que puede administrar los documentos electrónicos, es decir, todos aquellos documentos o archivos creados en los ordenadores, y, a menudo se almacenan en los sistemas de archivos locales de los usuarios. Los primeros sistemas de gestión de documentos electrónicos manejados ya sean los tipos de archivos de propiedad, o un número limitado de formatos de archivo.

Muchos de estos sistemas que posteriormente se conoció como los sistemas de imágenes de documentos, ya que se centró en la captura, almacenamiento, indexación y recuperación de los formatos de archivo de imagen. Sistemas de EDM evolucionaron a un punto donde los sistemas podrían administrar cualquier tipo de formato de archivo que podría almacenarse en la red.

Las aplicaciones crecieron para abarcar los documentos electrónicos, herramientas de colaboración, seguridad, flujo de trabajo y las capacidades de auditoría.

Estos sistemas permitieron una organización para capturar faxes y formularios, para guardar copias de los documentos, imágenes, y para almacenar los archivos de imagen en el repositorio para la seguridad y la recuperación rápida.

Mientras que muchos sistemas EDM almacenar documentos en su formato original, algunos sistemas de gestión de documentos basados en la web están comenzando a almacenar el contenido en forma de HTML. Estos sistemas de gestión de políticas requieren contenidos a ser importados en el sistema. Sin embargo, una vez que el contenido se importa, el software actúa como un motor de búsqueda para que los usuarios puedan encontrar lo que buscan más rápidamente.

El formato HTML permite una mejor aplicación de las capacidades de búsqueda, como búsqueda de texto completo y frenar.[1]

[1]http://centrodeartigos.com/articulos-utiles/article_107622.html

2. DESCRIPCIÓN DEL PROBLEMA

El centro de investigaciones de la Universidad Libre seccional Pereira no cuenta con una ayuda tecnológica para la gestión de procesos internos como son:

- Inscripción de proyectos de investigación
- Aprobación de propuestas
- Asignación de auxiliares de investigación
- Informes parciales
- Socialización de resultados
- Presentación de informes internos y externos.

Teniendo en cuenta la cantidad de solicitudes que maneja este departamento, se hace complicado llevar a cabo en un buen término de tiempo el cumplimiento de dichas solicitudes, todo el manejo de documentos se hace en físico sin ningún soporte digital para lo cual un proceso de auditoria se convierte en un verdadero reto para sus directores.

3. JUSTIFICACIÓN

El desarrollo e implementación de un aplicativo web para las actividades de investigación en la Facultad de Ingeniería de la Universidad Libre seccional Pereira, permitirá al Centro de Investigaciones de Ingeniería monitorear cada uno de los proyectos desarrollados por los investigadores.

Actualmente, estas actividades son monitoreadas mediante la presentación de informes físicos, o mediante conversaciones entre los investigadores y el director del Centro de Investigaciones, sin tener ningún soporte del desarrollo de estas actividades investigativas. La implementación del aplicativo web, como un sistema de información permitirá conocer.

Así mismo, servirá de medio de comunicación entre el Centro de Investigaciones y los investigadores para la presentación de propuestas y de informes, así como para realizar las solicitudes necesarias para adelantar los procesos investigativos.

Con el proyecto se quiere reducir drásticamente el manejo del papel, contribuyendo así con las políticas ambientales locales y nacionales.

4. OBJETIVOS

4.1. OBJETIVO GENERAL

Desarrollar un aplicativo web que permita monitorear los proyectos y productos de investigación que realizan los docentes investigadores de la facultad de ingeniería de la Universidad Libre seccional Pereira.

4.2. OBJETIVOS ESPECÍFICOS

- Definir las variables a incluir en el aplicativo mediante un análisis a los procedimientos utilizados en el centro de investigaciones.

- Diseñar el aplicativo web aplicando ingeniería de Software.

- Desarrollarel aplicativo webutilizando modelo cliente-vista-servidor y lenguajes de programación orientados a objetos como PHP y bases de datos como MySQL.

5. HIPOTESIS

El centro de investigaciones de la Universidad Libre seccional Pereira no cuenta con una ayuda tecnológica para la gestión de procesos internos.

Teniendo en cuenta la cantidad de solicitudes que maneja este departamento, se hace complicado llevar a cabo en un buen término de tiempo el cumplimiento de dichas solicitudes, todo el manejo de documentos se hace en físico sin ningún soporte digital para lo cual un proceso de auditoria se convierte en un verdadero reto para sus directores.

El presente proyecto pretende dar solución a la gestión de proyectos de investigación mencionada en el planteamiento del problema, mediante el desarrollo de un aplicativo web que permita mejorar los procesos de monitoreo y revisión de los productos de investigación de los docentes.

6. DELIMITACIÓN DEL PROYECTO

El alcance del proyecto está dirigido exclusivamente al Centro de Investigaciones de la Facultad de Ingeniería, permitiendo que su funcionamiento pueda realizarse sobre cualquier plataforma. Además se pretende implementar en dominio de la Universidad Libre Seccional Pereira.

El aplicativo web será desarrollado en la Universidad Libre Seccional Pereira y la población objetivo son los docentes investigadores, investigadores auxiliares y director de Centro de Investigaciones de la Facultad de Ingeniería de la Universidad Libre seccional Pereira.

El proyecto permite controles de seguridad sobre el manejo de los perfiles de los investigadores de la Facultad de Ingeniería de la Universidad Libre seccional Pereira.

7. MARCO REFERENCIAL

7.1. Marco teórico

A continuación se hace referencia a los fundamentos conceptuales, tecnológicos y metodológicos que orientan la elaboración del proyecto, así como una presentación de la relación que existe entre ellos en el contexto del proyecto.

7.1.1. Ingeniería de software

En el proyecto presente es la disciplina o área de la informática o ciencia de la computación, que ofrece conocimientos, técnicas y métodos para desarrollar y mantener software de calidad que resuelva problemas de todo tipo.

7.1.2. Ingeniería de requerimientos

En la ingeniería de sistemas y la ingeniería de software, la **Ingeniería de requisitos** o **Ingeniería de requerimientos** nos llevara a comprender todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para el desarrollo de un software, tomando en cuenta los diversos requisitos a satisfacer en el desarrollo de la aplicación web para la revisión y aprobación de los trabajos de investigación para el centro de investigaciones de universidad libre seccional Pereira.

7.1.3. RationalunifedProcess (RUP)

Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, que para este proyecto constituye la metodología estándar que se utilizara para el análisis, diseño, desarrollo, implementación y documentación de la aplicación web.

7.1.4. Extreme programming (XP)

Conocido como modelo de programación extrema, es un modelo utilizado en la ingeniería de software para el desarrollo de software de manera ágil, el cual será el modelo de programación utilizado en el proyecto, puesto que su ciclo de vida se acondiciona bastante bien para este tipo de software.

7.1.5. Metodología de la elicitación para la ingeniería de requerimientos.

En el contexto del presente proyecto es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos del problema formulado.

7.1.6. Requirements Management (REM)

Herramienta de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos del proyecto, se encuentra basada en la metodología definida en la Tesis Doctoral "Un Entorno Metodológico de Ingeniería de

Requisitos para Sistemas de Información", presentada por Amador Durán en septiembre de 2000.

7.1.7. UML

En esteproyecto será el lenguaje de modelado para especificar, construir, visualizar y documentar los artefactos del sistema de información a desarrollar, en el presente proyecto, un artefacto es la información utilizada o producida mediante un proceso de desarrollo de software.

7.1.8. Diseño de Software

Sera el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física.

7.1.9. Desarrollo de Software

Fase de la ingeniería de software o ingeniería de sistemas donde se concreta la fase de diseño, utilizando para ellos diversas metodologías de desarrollo y programación, tal como el desarrollo orientado a objetos, modelos cliente servidor, arquitecturas de bases de datos, de redes y lenguajes hipertextuales.

7.1.10. Lenguajes de programación

Para la codificación del software del proyecto se hace uso de lenguajes de programación, los cuales son idiomas artificiales diseñados para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras, pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana, está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa o software se le llama programación. Desarrollo Orientado a Objetos.

Se utilizara como metodología de programación, cuyo soporte fundamental será el objeto. Es un modo de trabajo más natural, que permite al desarrollador centrarse en solucionar el problema en lugar de tener que estar pensando en cómo decirle a la computadora que haga esto o lo otro.

7.1.11. HyperTextMarkupLanguage (HTML)

(«Lenguaje de etiquetas de hipertexto»), es el lenguaje de etiquetas predominante para la elaboración de páginas web, el cual se usará para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento.

Este lenguaje de etiquetas se utiliza en el presente proyecto para la visualización y gestión de los datos en el modelo cliente/servidor.

7.1.12. Apache

(Acrónimo de "a patchy server"). Servidor web de distribución libre y de código abierto, siendo el más popular del mundo desde abril de 1996, la aplicación permite ejecutarse en múltiples sistemas operativos como Windows, Novell NetWare, Mac OS X y los sistemas basados en Unix. Por ser de uso de libre y de código abierto será el servidor utilizado para este proyecto.

7.1.13. HyperTextPreprocessor (PHP)

Es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML, diseñado originalmente para la creación de páginas web dinámicas. Es el lenguaje de programación utilizado para el desarrollo de esta investigación.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (y de ese tipo, como Linux o Mac OS X) y Microsoft Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Permite la conexión a todo tipo de servidores de base de datos como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

7.1.14. Bases de datos

Una base de datos o banco de datos (en ocasiones abreviada con la sigla BD o con la abreviatura b. d.) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

7.1.15. MYSQL

El software MySQL proporcionara un servidor de base de datos SQL (StructuredQueryLanguage) veloz, multi-hilo, multiusuario y robusto. El servidor está proyectado tanto para sistemas críticos en producción soportando intensas cargas de trabajo como para empotrarse en sistemas de desarrollo masivo de software. El software MySQL tiene licencia dual, pudiéndose usar de forma gratuita bajo licencia GNU o bien adquiriendo licencias comerciales de MySQL AB en el caso de no desear estar sujeto a los términos de la licencia GPL. MySQL es una marca registrada de MySQL AB.

7.1.16. Implementación

Es el proceso donde se instalara el aplicativo nuevo, como resultado de un análisis y diseño previo de la situación o mejoramiento de la forma de llevar a cabo un proceso automatizado. Al implementar un sistema lo primero que debemos hacer es asegurarnos que el sistema sea operacional o que funcione de acuerdo a los requerimientos del análisis y permitir que los usuarios puedan operarlos.

7.1.17. Pruebas

Durante este proceso se debe poner en práctica todas las estrategias posibles para garantizar que el usuario inicial del sistema se encuentre libre de problemas.

7.1.18. Gestión y control de la calidad

Son las técnicas y actividades de carácter operativo, que se utilizaran para satisfacer los requisitos relativos a la calidad, centrados en dos objetivos fundamentales:

- Mantener bajo control un proceso.
- eliminar las causas de los defectos en las diferentes fases del ciclo de vida.

7.2. Marco conceptual

7.2.1. Ingeniería de software

Según la definición del IEEE, citada por (Lewis 1994) "**software** es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo". Según el mismo autor, "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software (SE del inglés *Software Engineering*) es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la **Ingeniería de Software** es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y costo-efectivos" (Cota 1994).

El **proceso de ingeniería de software** se define como "un conjunto de etapas parcialmente ordenadas con la intención de logra un objetivo, en este caso, la obtención de un producto de software de calidad" (Jacobson 1998).El **proceso de desarrollo de software** "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Jacobson 1998).

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el **ciclo de vida del software** que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Actualmente se encuentra en una etapa de madurez el enfoque Orientado a Objetos (OO) como paradigma del desarrollo de sistemas de información. El Object Management Group (OMG) es un consorcio a nivel internacional que integra a los principales representantes de la industria de la tecnología de información OO. El OMG tiene como objetivo central la promoción, fortalecimiento e impulso de la industria OO. El OMG propone y adopta por consenso especificaciones entorno a la tecnología OO. Una de las especificaciones más importantes es la adopción en 1998 del Lenguaje de Modelado Unificado o UML (del inglés *UnifiedModelingLanguage*) como un estándar, que junto con el Proceso Unificado están consolidando la tecnología OO.[2]

7.2.2. Ingeniería de requerimientos

El proceso de recopilar, analizar y verificar las necesidades del cliente o usuario para un sistema es llamado ingeniería de requerimientos. La meta de la ingeniería de requerimientos (IR) es entregar una especificación de requisitos de software correcta y completa. Algunos otros conceptos de ingeniería de requerimientos son:

[2]http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

“Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software”. (Pressman, 2006: 155)

“La ingeniería de requerimientos es el proceso de desarrollar una especificación de software. Las especificaciones pretender comunicar las necesidades del sistema del cliente a los desarrolladores del sistema”. (Sommerville, 2005: 82)

En síntesis, el proceso de ingeniería de requerimientos se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto de software determinado, donde es muy importante tomar en cuenta que el aporte de la IR vendrá a ayudar a determinar la viabilidad de llevar a cabo el software (si es factible llevarlo a cabo o no),pasando posteriormente por un subproceso de obtención y análisis de requerimientos, su especificación formal, para finalizar con el subproceso de validación donde se verifica que los requerimientos realmente definen el sistema que quiere el cliente.

7.2.3. RationalunifedProcess (RUP)

El **Proceso Unificado** "es un proceso de desarrollo de software configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos en el desarrollo de software de misión crítica en una variedad de industrias por la compañía Rational", donde confluyen 'los tres amigos' como se

llaman a sí mismos o los tres grandes OO: Grady Booch, James Rumbaugh e Ivar Jacobson (M&R 1998).

El Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

7.2.3.1. El ciclo de vida del software en el Proceso Unificado

Las **fases del ciclo de vida del software** son: concepción, elaboración, construcción y transición. La concepción es definir el alcance del proyecto y definir el caso de uso. La elaboración es proyectar un plan, definir las características y cimentar la arquitectura. La construcción es crear el producto y la transición es transferir el producto a sus usuarios (Booch 1998).

7.2.4. Extreme Programming (XP)

Extreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.)

aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías "clásicas" es que son demasiado burocráticas. Hay tanto que hacer para seguir la metodología que, a veces, el ritmo entero del desarrollo se retarda. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de "Metodologías Ágiles". Estos nuevos métodos buscan un punto medio entre la ausencia de procesos y el abuso de los mismos, proponiendo un proceso cuyo esfuerzo valga la pena.

XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos recientes. La metodología propuesta en XP está diseñada para entregar el software que los clientes necesitan en el momento en que lo necesitan. XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aún en fases tardías del ciclo de vida del desarrollo[3].

La metodología también enfatiza el trabajo en equipo. Tanto gerentes como clientes y desarrolladores son partes del mismo equipo dedicado a entregar software de calidad.

XP fue introducida como metodología ágil de desarrollo de software sobre finales de los 1990s. Uno de los conocidos "caso de éxito" fue publicado a finesde 1998, cuando Kent Beck[4] introdujo la nueva metodología en el proyecto de desarrollo denominado C3 (Chrysler ComprehensiveCompensation) para la firma Chrysler[5].

[3] What is Extreme Programming? http://www.extremeprogramming.org/what.html
[4] Kent Beck. http://www.threeriversinstitute.org/Kent%20Beck.htm
[5] Chrysler Goes to "Extremes", Distributed Computing, October 1998. Kent Beck et. al. (The C3 Team)

7.2.4.1. Ciclo de vida de Software en XP

Para apreciar los conceptos del ciclo de desarrollo de software en XP introduciremos brevemente los conceptos principales de las metodologías de desarrollo de software tradicionales[6].

7.2.4.1.1. Modelo en cascada

El modelo de cascada tiene sus orígenes en la década de 1970[7], y se define como una secuencia de actividades bien planificadas y estructuradas. El proceso distingue claramente las fases de especificación de las de desarrollo y éstas, a su vez, de las de testing. Es, seguramente, la metodología más extendida y utilizada.

Este modelo se basa fuertemente en que cada detalle de los requisitos se conoce de antemano, previo de comenzar la fase de codificación o desarrollo, y asume, además, que no existirán cambios significativos en los mismos a lo largo del ciclo de vida del desarrollo.

7.2.4.1.2. Modelo incremental

El modelo incremental consiste en un desarrollo inicial de la arquitectura completa del sistema, seguido de sucesivos incrementos funcionales. Cada incremento tiene su propio ciclo de vida y se basa en el anterior, sin cambiar su

[6] Ingeniería de Software orientada a objetos con UML Java e Internet, Alfredo Weitzenfeld Thomson editores, 2005.ISBN 970-686-190-4

[7] Managing the development of large software systems.Dr. Winston W Royce, Proceedings, IEEE Wescon. Pages 1-9, August 1970. www.cs.umd.edu/class/spring2003/ cmsc838p/Process/waterfall.pdf

funcionalidad ni sus interfaces. Una vez entregado un incremento, no se realizan cambios sobre el mismo, sino únicamente corrección de errores. Dado que la arquitectura completa se desarrolla en la etapa inicial, es necesario, al igual que en el modelo en cascada, conocer los requerimientos completos al comienzo del desarrollo.

Respecto al modelo en cascada, el incremental tiene la ventaja de entregar una funcionalidad inicial en menor tiempo.

7.2.4.1.3. Modelo evolutivo

El modelo evolutivo es, en cierta forma, similar al incremental, pero admite que la especificación no esté completamente determinada al comienzo del ciclo de vida.

Los requerimientos que estén suficientemente detallados al comienzo darán lugar a una entrega inicial, mientras que los siguientes incrementos serán cambios progresivos que implementen “deltas” de especificación de requerimientos. El modelo admite que, si la especificación no es suficientemente clara al principio, puede desarrollarse un prototipo experimental, que tiene como función validar o identificar los requisitos del sistema.

7.2.4.1.4. Modelo espiral

El modelo de espiral, introducido por Barry Bohem a fines de la década de 1980[8], intenta combinar las ventajas del modelo en cascada con el modelo evolutivo. El modelo enfatiza el estudio de los riesgos del proyecto, como por ejemplo las especificaciones incompletas. Se prevé, en este modelo, varios ciclos o "vueltas de espiral", cada uno de ellos con cuatro etapas: Definición de objetivos, Evaluación y reducción del riesgo, Desarrollo y validación y Planificación del siguiente ciclo. En este modelo, una actividad comienza solo cuando se entienden los objetivos y riesgos involucrados. El desarrollo se incrementa en cada etapa, generando una solución completa. La metodología en espiral ha sido utilizada con éxito en grandes sistemas, pero su complejidad la hace desaconsejable para el desarrollo de sistemas medianos o pequeños.

7.2.4.1.5. Modelo XP

La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres. Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. Este modelo es analizado por Kent Beck (1999)[9] en donde propone las ventajas de un contrato con alcances opcionales.

[8] A Spiral Model of Software Development and Enhancement, Barry W Boehm, Computer (IEEE), Volume 21, Issue 5, May 1988 Pages 61 - 72
[9] Optional Scope Contracts, Kent Beck, Dave Cleal, 1999

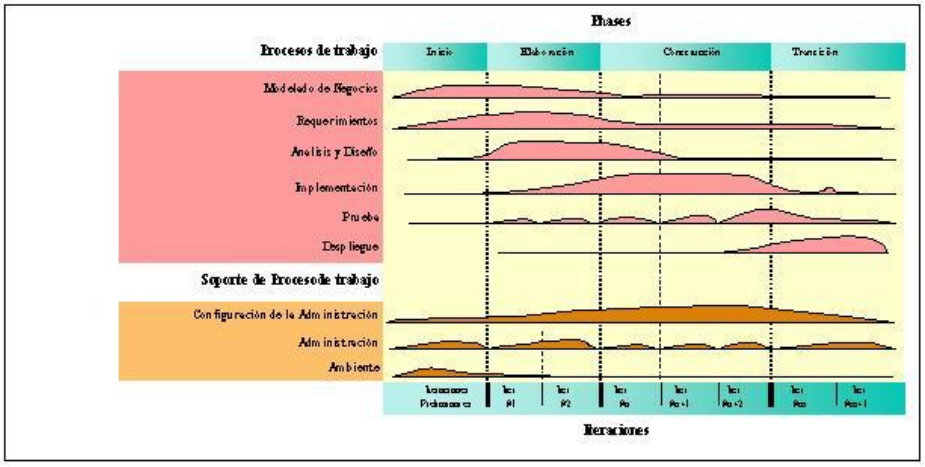
Como se detalló en los apartados anteriores, los ciclos de vida “tradicionales” proponen una clara distinción entre las etapas del proyecto de software, y tienen un plan bien preestablecido acerca del proceso de desarrollo. Asimismo, en todos ellos se parte de especificaciones claras, si no del total del proyecto, por lo menos de una buena parte inicial.

El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.

Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP (y que serán detalladas más adelante).

Típicamente un proyecto con XP lleva 10 a 15 ciclos o iteraciones. La siguiente ilustración esquematiza los ciclos de desarrollo en cascada e iterativos tradicionales (por ejemplo, incremental o espiral), comparados con el de XP.

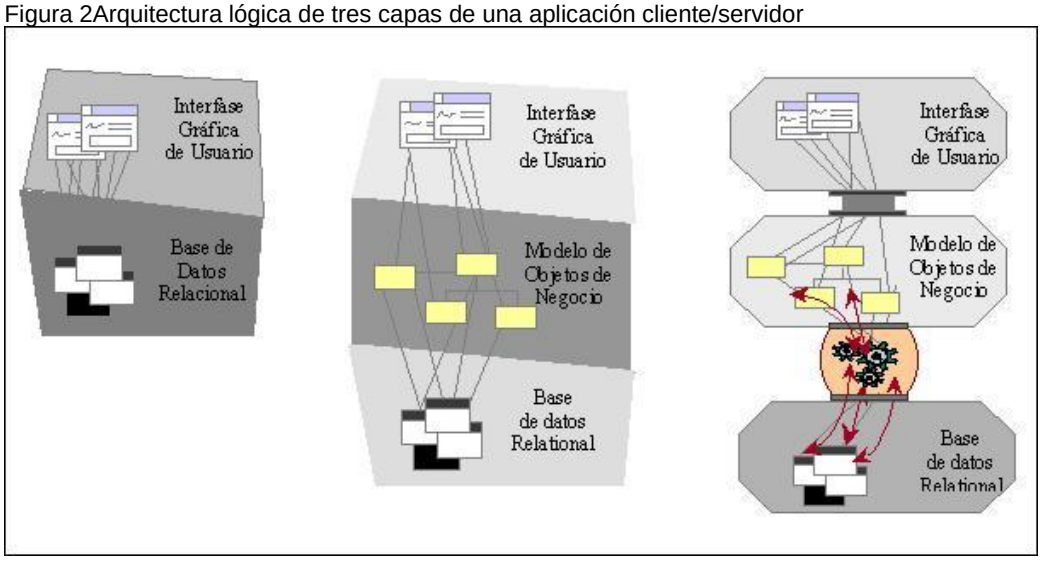
Figura 1Estructura del Proceso Unificado



Fuente: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

Según (Microsoft 1997), el diseño de software se realiza a tres niveles: conceptual, lógico y físico.

Figura 2Arquitectura lógica de tres capas de una aplicación cliente/servidor

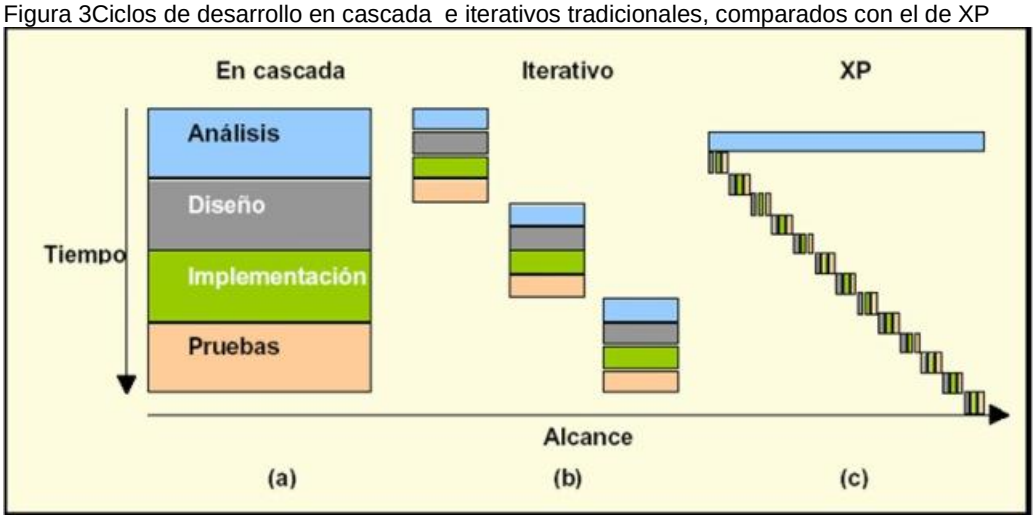


Fuente: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

Un ingeniero de software necesita de herramientas, entre ellas las herramientas de Rational son las más avanzadas, pero son muy costosas. También puede utilizar las herramientas de oficina como un editor de textos, un

modelador de datos, etc., muchas de ellas son de código abierto y aún están de desarrollo. Utiliza las que más te sean de utilidad.[10]

Figura 3Ciclos de desarrollo en cascada e iterativos tradicionales, comparados con el de XP



Fuente: Extreme Programming (XP): un nuevo método de desarrollo de software, César F. Acebal, Juan M. Cueva Lovelle, Depto. de Informática, Área de Lenguajes y Sistemas Informáticos, Universidad de Oviedo. NOVATICA, Marzo Abril 2002, pp 8-12

Si bien el ciclo de vida de un proyecto XP es muy dinámico, se puede separar en fases[11]. Varios de los detalles acerca de las tareas de éstas fases se detallan más adelante, en la sección “Reglas y Practicas”:

7.2.5. Elicitación

“La Elicitación de Requisitos –ER– es la piedra angular en el desarrollo de proyectos software y tiene un impacto muy alto en el diseño y en las demás fases del ciclo de vida del producto. Si se realiza apropiadamente, puede ayudar a reducir los cambios y las correcciones en los requisitos. Además, la calidad de la elicitación determina la exactitud de la retroalimentación al cliente

[10] http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html
[11] XP: A Project Manager’s Primer, Steward Baird, March 22, 2002. Prentice Hall, Professional Technical Reference. http://www.phptr.com/articles/article.asp?p=26060&seqNum=6&rl=1

acerca de la integridad y validez de los requisitos. Debido a que esta fase es crítica y de alto impacto en el proyecto, es muy importante que la labor de elicitar se realice lo más cercano posible a la "perfección". Teniendo en cuenta las diferentes características de los proyectos software, en este trabajo se proponen algunas reglas generales para llevar a cabo la ER con base en la discusión y en la explicación de los procesos relacionados y métodos aplicados en los diferentes tipos de proyectos software."[12]

7.2.6. UML

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.

[12] M. Manies& U. Nikual. "La Elicitación de Requisitos en el contexto de un proyecto software". Ing.USBMed, Vol. 2, No. 2, pp. 25-29. ISSN: 2027-5846. Jul-Dic, 2011.

- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997. Hubo otras cinco propuestas rivales. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997. El OMG llama a este documento OMG UML versión 1.1. El OMG está actualmente en proceso de mejorar una edición técnica de esta especificación, prevista su finalización para el 1 de abril de 1999.

7.2.6.1. UML ofrece notación y semántica estándar

UML prescribe una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Previamente, un diseño orientado

a objetos podría haber sido modelado con cualquiera de la docena de metodologías populares, causando a los revisores tener que aprender las semánticas y notaciones de la metodología empleada antes que intentar entender el diseño en sí. Ahora con UML, diseñadores diferentes modelando sistemas diferentes pueden sobradamente entender cada uno los diseños de los otros.

7.2.6.2. UML no es un Método

Aun así, UML no prescribe un proceso o método estándar para desarrollar un sistema. Hay varias metodologías existentes; entre las más populares se incluyen las siguientes:

- Catalysis: Un método orientado a objetos que fusiona mucho del trabajo reciente en métodos orientados a objetos, y además ofrece técnicas específicas para modelar componentes distribuidos.
- Objetory: Un método de Caso de Uso guiado para el desarrollo, creado por Ivar Jacobson.
- Shlaer/Mellor: El método para diseñar sistemas de tiempo real, puesto en marcha por Sally ShlaeryStevenMellor en dos libros de 1991, Ciclos de vida de Objetos, modelando el Mundo en Estados y Ciclos de vida de Objetos, Modelando el mundo en Datos (Prentice Hall). Shlaer/Mellor continúan actualizando su método continuamente (la actualización más reciente es el OOA96 report), y recientemente publicaron una guía sobre cómo usar la notación UML con Shlaer/Mellor.
- Fusión: Desarrollado en Hewlett Packard a mediados de los noventa como primer intento de un método de diseño orientado a objetos estándar. Combina OMT y Booch con tarjetas CRC y métodos formales[13].

[13] www.hpl.hp.com/fusion/file/teameps.pdf

- OMT: La Técnica de Modelado de Objetos fue desarrollada por James Rumbaugh y otros, y publicada en el libro de gran influencia "Diseño y Modelado Orientado a Objetos" (Prentice Hall, 1991). Un método que propone análisis y diseño 'iterative', más centrado en el lado del análisis.
- Booch: Parecido al OMT, y también muy popular, la primera y segunda edición de "Diseño Orientado a Objetos, con Aplicaciones" (Benjamin Cummings, 1991 y 1994), (Object-OrientedDesign, WithApplications), detallan un método ofreciendo también diseño y análisis 'iterative', centrándose en el detallado del diseño.

Además, muchas organizaciones han desarrollado sus propias metodologías internas, usando diferentes diagramas y técnicas con orígenes varios. Ejemplos son el método Catalyst por ComputerSciencesCorporation (CSC) o el WorlwideSolutionDesign and DeliveryMethod (WSDDM) por IBM. Estas metodologías difieren, pero generalmente combinan análisis de flujo de trabajo, captura de los requisitos, y modelado de negocio con modelado de datos, con modelado de objetos usando varias notaciones(OMT, Booch, etc), y algunas veces incluyendo técnicas adicionales de modelado de objetos como Casos de Uso y tarjetas CRC. La mayoría de estas organizaciones están adoptando e incorporando el UML como la notación orientada a objetos de sus metodologías.

Algunos modeladores usarán un subconjunto de UML para modelar 'whatthey'reafter', por ejemplo simplemente el diagrama de clases, o solo los diagramas de clases y de secuencia con Casos de Uso. Otros usarán una suite más completa, incluyendo los diagramas de estado y actividad para modelar sistemas de tiempo real, y el diagrama de implementación para modelar sistemas distribuidos. Aun así, otros no estarán satisfechos con los diagramas ofrecidos por UML, y necesitarán extender UML con otros diagramas como modelos relacionales de datos y 'CRC cards'.

7.2.6.3. Una perspectiva general de UML

7.2.6.3.1. Una vuelta por un caso de uso

Una vez más, UML es una notación, no un método. No prescribe un proceso para modelar un sistema. No obstante, como UML incluye los diagramas de casos de uso, se le considera estar dotado de una aproximación al diseño centrada en el problema con los casos de uso. El Diagrama de Caso de Uso nos da el punto de entrada para analizar los requisitos del sistema, y el problema que necesitamos solucionar.

7.2.6.3.2. Casos de Uso y Diagramas de Interacción

Un caso de uso se modela para todos los procesos que el sistema debe llevar a cabo. Los procesos se describen dentro del caso de uso por una descripción textual o una secuencia de pasos ejecutados. Los Diagramas de Actividad se pueden usar también para modelar escenarios gráficamente. Una vez que el comportamiento del sistema está captado de esta manera, los casos de uso se examinan y amplían para mostrar qué objetos se interrelacionan para que ocurra este comportamiento. Los Diagramas de Colaboración y de Secuencia se usan para mostrar las relaciones entre los objetos.

7.2.6.3.3. Clases y Diagramas de Implementación

Conforme se van encontrando los objetos, pueden ser agrupados por tipo y clasificados en un Diagrama de Clase. Es el diagrama de clase el que se

convierte en el diagrama central del análisis del diseño orientado a objetos, y el que muestra la estructura estática del sistema. El diagrama de clase puede ser dividido en capas: aplicación, y datos, las cuales muestran las clases que intervienen con la interfaz de usuario, la lógica del software de la aplicación, y el almacenamiento de datos respectivamente. Los Diagramas de Componentes se usan para agrupar clases en componentes o módulos. La distribución general del hardware del sistema se modela usando el Diagrama de Implementación.

7.2.6.3.4. Tarjetas CRC (CRC cards) - Una extensión informal de UML

Como una extensión informal a UML, la técnica de las tarjetas CRC se puede usar para guiar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades.

7.2.6.3.5. Diagramas de Estado

El comportamiento en tiempo real de cada clase que tiene comportamiento dinámico y significativo, se modela usando un Diagrama de Estado. El diagrama de actividad puede ser usado también aquí, esta vez como una extensión del diagrama de estado, para mostrar los detalles de las acciones llevadas a cabo por los objetos en respuesta a eventos internos. El diagrama de actividad se puede usar también para representar gráficamente las acciones de métodos de clases.

7.2.6.3.6. Implementando el diseño

La implementación del sistema trata de traducir información desde múltiples modelos UML en código y estructura de bases de datos. Cuando se modela un sistema grande, es útil fragmentar el sistema en su capa ’business’ (incluyendo los objetos de la interfaz de usuario), su capa de aplicación (incluyendo los objetos de implementación), y su capa de datos (incluyendo la estructura de la base de datos y el acceso a objetos).

7.2.6.3.7. Implementando la aplicación

El Diagrama de Clase se usa para generar una estructura base del código en el lenguaje escogido. Información de los diagramas de interacción, estado, y actividad, puede ofrecer detalles de la parte procedimental del código de implementación.

7.2.6.3.8. Implementando el diseño de Bases de Datos

La capa de datos del diagrama de clase se puede usar para implementar directamente un diseño orientado a objetos de una base de datos, o, como extensión de UML, puede ser referenciado en un diagrama de relación de entidad para más análisis de relaciones de entidad. Está en el diagrama de relación de entidad (ER diagram, entityrelationship) el cual relaciona entre entidades que pueden ser modeladas basadas en atributos clave. El diagrama de relación de entidad lógico ofrece una base desde la cual construir un diagrama físico representando las tablas y relaciones actuales de la base de datos relacional.

7.2.6.3.9. Probar teniendo en cuenta los requisitos

Los casos de uso se utilizan también para probar el sistema y ver si satisface los requisitos iniciales. Los pasos de los casos de uso van llevando a cabo para determinar si el sistema está satisfaciendo los requisitos del usuario.

7.2.6.3.10. Modelado de Casos de Uso

El modelado de Casos de Uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione. No es realmente una aproximación a la orientación a objetos; es realmente una forma de modelar procesos. Es, sin embargo, una manera muy buena de dirigirse hacia el análisis de sistemas orientado a objetos. Los casos de uso son generalmente el punto de partida del análisis orientado a objetos con UML.

El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. Se dibujan como "muñecos" de palo. Actualmente representan el tipo de usuario, no una instancia de usuario. Los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. Se dibujan como elipses.

Cada caso de uso se documenta por una descripción del escenario. La descripción puede ser escrita en modo de texto o en un formato paso a paso. Cada caso de uso puede ser también definido por otras propiedades, como las

condiciones pre - y post - del escenario - condiciones que existen antes de que el escenario comience, y condiciones que existen después de que el escenario se completa. Los Diagramas de Actividad ofrecen una herramienta gráfica para modelar el proceso de un Caso de Uso.

7.2.6.3.11. Diagramas de Secuencia

El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista ’business’ del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal delos objetos es arbitraria.

Durante el análisis inicial, el modelador típicamente coloca el nombre ’business’ de un mensaje en la línea del mensaje. Más tarde, durante el diseño, el nombre ’business’ es reemplazado con el nombre del método que está siendo llamado por un objeto en el otro. El método llamado, o invocado, pertenece a la definición de la case instanciada por el objeto en la recepción final del mensaje.

7.2.6.3.12. Diagramas de Colaboración

El Diagrama de Colaboración presenta una alternativa al diagrama de secuencia para modelar interacciones entre objetos en el sistema. Mientras que el diagrama de secuencia se centra en la secuencia cronológica del escenario que estamos modelando, el diagrama de colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario. Los objetos se conectan por medio de enlaces, cada enlace representa una instancia de una asociación entre las clases implicadas. El enlace muestra los mensajes enviados entre los objetos, el tipo de mensaje (sincrónico, asincrónico, simple, blanking, y ’time-out’), y la visibilidad de un objeto con respecto a los otros.

7.2.6.3.13. Análisis y Diseño con el Diagrama de Clase

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifíca para satisfacer los detalles de las implementaciones.

7.2.6.3.14. Modelando el comportamiento de las Clases con Diagramas de Estado

Mientras los diagramas de interacción y colaboración modelan secuencias dinámicas de acción entre grupos de objetos en un sistema, el diagrama de estado se usa para modelar el comportamiento dinámico de un objeto en particular, o de una clase de objetos.

Un diagrama de estado se modela para todas las clases que se consideran con un comportamiento dinámico. En él, modelas la secuencia de estado que un objeto de la clase atraviesa durante su vida en respuesta a los estímulos recibidos, junto con sus propias respuestas y acciones. Por ejemplo, un comportamiento de un objeto se modela en términos de en qué estado está inicialmente, y a qué estado cambia cuando recibe un evento en particular. También modelas qué acciones realiza un objeto en un estado en concreto.

Los estados representan las condiciones de objetos en ciertos puntos en el tiempo. Los eventos representan incidentes que hacen que los objetos pasen de un estado a otro. Las líneas de transición describen el movimiento desde un estado hasta otro. Cada línea de transición se nombre con el evento que causa esta transición. Las acciones ocurren cuando un objeto llega a un estado.

7.2.6.3.15. Diagramas de Actividad

El Diagrama de Actividad es un diagrama de flujo del proceso multi-propósito que se usa para modelar el comportamiento del sistema. Los diagramas de

actividad se pueden usar para modelar un Caso de Uso, o una clase, o un método complicado.

Un diagrama de actividad es parecido a un diagrama de flujo; la diferencia clave es que los diagramas de actividad pueden mostrar procesado paralelo (parallelprocessing). Esto es importante cuando se usan diagramas de actividad para modelar procesos 'bussiness' algunos de los cuales pueden actuar en paralelo, y para modelar varios hilos en los programas concurrentes.

7.2.6.3.16. Modelando Componentes de Software

El Diagrama de Componentes se usa para modelar la estructura del software, incluyendo las dependencias entre los componentes de software, los componentes de código binario, y los componentes ejecutables. En el Diagrama de Componentes modelas componentes del sistema, a veces agrupados por paquetes, y las dependencias que existen entre componentes (y paquetes de componentes).

7.2.6.3.17. Modelando la Distribución y la Implementación

Los Diagramas de Implementación se usan para modelar la configuración de los elementos de procesado en tiempo de ejecución (run-time processingelements) y de los componentes, procesos y objetos de software que viven en ellos. En el diagrama 'deployment', empiezas modelando nodos físicos y las asociaciones de comunicación que existen entre ellos. Para cada nodo, puedes indicar qué instancias de componentes viven o corren (se

ejecutan) en el nodo. También puedes modelar los objetos que contiene el componente.
Los Diagramas de Implementación se usan para modelar sólo componentes que existen como entidades en tiempo de ejecución; no se usan para modelar componentes solo de tiempo de compilación o de tiempo de enlazado. Puedes también modelar componentes que migran de nodo a nodo u objetos que migran de componente a componente usando una relación de dependencia con el estereotipo ’becomes’(se transforma)

7.2.6.3.18. Diseño de Bases de Datos Relacionales -- Una extensión informal de UML

El Diagrama de Clase presenta un mecanismo de implementación neutral para modelar los aspectos de almacenado de datos del sistema. Las clases persistentes, sus atributos, y sus relaciones pueden ser implementados directamente en una base de datos orientada a objetos. Aun así, en el entorno de desarrollo actual, la base de datos relacional es el método más usado para el almacenamiento de datos.

Es en el modelado de esta área donde UML se queda corto. El diagrama de clase de UML se puede usar para modelar algunos aspectos del diseño de bases de datos relacionales, pero no cubre toda la semántica involucrada en el modelado relacional, mayoritariamente la noción de atributos clave que relacionan entre sí las tablas unas con otras. Para capturar esta información, un Diagrama de Relación de Entidad (ER diagram) se recomienda como extensión a UML.

El Diagrama de Clase se puede usar para modelar la estructura lógica de la base de datos, independientemente de si es orientada a objetos o relacional, con clases representando tablas, y atributos de clase representando columnas. Si una base de datos relacional es el método de implementación escogido, entonces el diagrama de clase puede ser referenciado a un diagrama de relación de entidad lógico. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos; el modelador dispone de varias opciones sobre cómo inferir asociaciones en relaciones entre entidades. Las relaciones de herencia son referenciadas directamente a súper-sub relaciones entre entidades en un diagrama de relación de entidad (ER diagram).

Ya en el Diagrama de Relación de Entidad, el modelador puede empezar el proceso de determinar cómo el modelo relacional encaja; y qué atributos son claves primarias, claves secundarias, y claves externas basadas en relaciones con otras entidades. La idea es construir un modelo lógico que sea conforme a las reglas de normalización de datos. Al implementar el diseño relacional, es una estrategia encaminada a hacer referencia al diagrama de relación de entidad lógico a un diagrama físico que represente el objetivo, el RDBMS. El diagrama físico puede ser des normalizado para lograr un diseño de base de datos que tiene tiempos eficientes de acceso a los datos. Las relaciones súper-sub entre entidades se resuelven por las estructuras de tablas actuales.

Además, el diagrama físico se usa para modelar propiedades específicas de cada fabricante para el RDBMS. Se crean varios diagramas físicos si hay varios RDBMSs siendo ’deployed’; cada diagrama fisco representa uno de los RDBMS que son nuestro objetivo.

7.2.7. Diseño de Software

A través de la historia de la ingeniería del software ha evolucionado un conjunto de conceptos fundamentales de diseño de software, aunque el grado de interés en cada concepto ha variado con los años, han pasado la prueba del tiempo ofreciendo cada uno al ingeniero de software fundamentos sobre el cual pueden aplicarse métodos de diseño más elaborados.

El diseño de Software juega un papel importante en el desarrollo de software lo cual permite al ingeniero de software producir varios modelos del sistema o producto de que se va a construir el mismo que forman una especie de plan de la solución de la aplicación. Estos modelos pueden evaluarse en relación con su calidad y mejorarse antes de generar código, de realizar pruebas y de que los usuarios finales se vean involucrados a gran escala. El diseño es el sitio en el que se establece la calidad del software.

Diseño es definido como: ***"El proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o componente que resulta de este proceso" (IEEE610.12-90).***

Palabras Claves

Definición de Documentos de Software (IEEE)

SQAP: Software Quality Assurance Plan IEEE 730

SCMP: Software Configuration Management Plan IEEE 828

STD: Software Test Documentation IEEE 829

SRS: Software Requirements Specification IEEE 830

SVVP: Software Validation & Verification Plan IEEE 1012

SDD: Software Design Description IEEE 1016

SPMP: Software Project Management Plan IEEE 1058

"***El milagro más común de la ingeniería de software es la transición del análisis al diseño y del diseño al código***" **Richard Due**

7.2.7.1. Proceso del Diseño de Software

7.2.7.1.1. Diseño Arquitectónico

El diseño arquitectónico puede representarse al usar uno o más de muchos modelos diferentes. Los modelos estructurales representan la arquitectura como una colección organizada de componentes del programa. Los modelos del marco de trabajo repetible incrementan el grado de abstracción del diseño al intentar identificar marcos de trabajo repetibles del diseño arquitectónico que se encuentran en tipos de aplicaciones similares.

7.2.7.1.2. Cohesión

Es una extensión natural del concepto de ocultamiento de la información. Un módulo con cohesión realiza una sola tarea dentro de un procedimiento de software, requiriendo poca interacción con los procedimientos que se realizan en otras partes del programa. Un módulo con cohesión debería hacer una sola cosa.Siempre debemos buscar la cohesión más alta, aunque la parte media del espectro es a menudo aceptable.

Coincidencialmente cohesivo: un módulo que realiza un conjunto de tareas poco relacionadas las unas con las otras.

Cohesión lógica: realiza tareas relacionadas lógicamente (produce todas las salidas).

Cohesión temporal: contienen tareas relacionadas por el hecho de que todas deben hacerse en el mismo intervalo de tiempo.

Cohesión procedimental: cuando los elementos de procesamiento están relacionados y deben ejecutarse en un orden específico.

Cohesión de comunicación: todos los elementos de procesamiento se concentran en un área de la estructura de datos.

7.2.7.1.3. La descomposición y la modularización

Los patrones de arquitectura y diseño de software materializan la modularidad; es decir, el software se divide en componentes con nombres independientes y que es posible abordar en forma individual. Estos componentes llamados módulos se integran para satisfacer los requisitos del problema.

7.2.7.1.4. Modularidad

Se divide el software en componentes identificables y tratables por separado, denominados módulos, que están integrados para satisfacer los requisitos del programa. Hay un número m de módulos que resultarían en un costo de desarrollo mínimo, pero no tenemos la sofisticación necesaria para predecir con seguridad

7.2.7.1.5. Temas Claves en el Diseño de Software

A la hora de diseñar software hay una serie de cuestiones fundamentales que se deben tomar en cuenta. Algunos relacionados con la calidad así como los concernientes a la dirección como por ejemplo el rendimiento. Además de cómo se descomponen, organizan los paquetes de los componentes de software. Esto es tan fundamental que en todo el proceso de diseño que se debe abordar de una manera u otra.

"(Aspectos) no suelen ser unidades de descomposición funcional del software, sino más bien a las propiedades que afectan eldesempeño o la semántica de los componentes en el sistema en diferentes maneras " (Kic97).

7.2.7.1.6. Concurrencia

La forma de descomponer el software en los procesos, tareas e hilos tratar relacionarlos con la eficiencia, la atomicidad, la sincronización, y demás cuestiones de programación.

7.2.7.1.7. Control y manejo de Eventos

Cómo organizar los datos y el controlar el flujo, manejo de reactivo y temporal de los acontecimientos a través de diversos mecanismos, tales como la invocación implícita de llamadas y sus intentos.

7.2.7.1.8. Distribución de Componentes

Cómo distribuir el software en el hardware, cómo los componentes se comunican, cómo se puede usar una plataforma al utilizarse para hacer frente a software heterogéneos.

7.2.7.1.9. Error y Gestión de Excepciones Tolerancia a Fallos

El análisis y la gestión del riesgo son una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre.

Un riesgo es un problema potencial que puede ocurrir o no. Pero sin tener en cuenta el resultado, realmente es una buena idea es identificarlo, evaluar su probabilidad de aparición, estimar su impacto, y establecer un plan de contingencia por si ocurre el problema.

Una estrategia considerablemente más inteligente para el control del riesgo es ser proactivo. La estrategia proactiva empieza mucho antes de que comiencen los trabajos técnicos. Se identifican los riesgos potenciales, se evalúa su probabilidad y su impacto y se establece una prioridad según su importancia. Después, el equipo de Software establece un plan para controlar el riesgo. El primer objetivo es evitar el riesgo, pero como no se pueden evitar todos los riesgos, el equipo trabaja para desarrollar un plan de contingencia que le permita responder de una manera eficaz y controlada.

7.2.7.1.10. Estructura y Arquitectura de Software

En el sentido estricto, una arquitectura de software es "Una descripción de los subsistemas y componentes de un sistema de software y las relaciones que existen entre ellos" (Bus96: c6).

A mediados de 1990, la arquitectura empezó a emerger como una disciplina más amplia que implica el estudio de las estructuras y las arquitecturas de software en una forma más genérica, dando ideas interesantes sobre diseño del software en diferentes niveles de abstracción.

Algunos de estos conceptos son muy útiles durante el diseño arquitectónico (estilo de arquitectura), de software específico, así como en su diseño de

detalle (nivel inferior, patrones de diseño). Así también para el diseño de sistemas genéricos lo que lleva a la concepción de las familias de los programas(conocidas como líneas de productos). La mayoría de estos conceptos pueden verse como intentos de describir, por tanto la reutilización del diseño genérico del conocimiento.

El desarrollo de un sistema con gran cantidad de software requiere que este sea visto desde diferentes perspectivas. Diferentes usuarios (usuario final, analistas, desarrolladores, integradores, jefes de proyecto...) siguen diferentes actividades en diferentes momentos del ciclo de vida del proyecto, lo que da lugar a las diferentes vistas del proyecto, dependiendo de qué interese más en cada instante de tiempo.

7.2.7.1.11. La vista de casos de uso

Comprende la descripción del comportamiento del sistema tal y como es percibido por los usuarios finales, analistas y encargados de las pruebas y se utilizan los diagramas de casos de uso para capturar los aspectos estáticos mientras que los dinámicos son representados por diagramas de interacción, estados y actividades.

7.2.7.1.12. La vista de diseño

Comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y de la solución. Esta vista soporta principalmente los requisitos funcionales del sistema, o sea, los servicios que el sistema debe proporcionar.

Los aspectos estáticos se representan mediante diagramas de clases y objetos y los aspectos dinámicos con diagramas de interacción, estados y actividades.

7.2.7.1.13. La vista de procesos

Comprende los hilos y procesos que forman mecanismos de sincronización y concurrencia del sistema cubriendo el funcionamiento, capacidad de crecimiento y el rendimiento del sistema. Con UML, los aspectos estáticos y dinámicos se representan igual que en la vista de diseño, pero con el énfasis que aportan las clases activas, las cuales representan los procesos y los hilos.

7.2.7.1.14. La Vista de implementación

Comprende los componentes y los archivos que un sistema utiliza para ensamblar y hacer disponible el sistema físico. Se ocupa principalmente de la gestión de configuraciones de las distintas versiones del sistema. Los aspectos estáticos se capturan con los diagramas de componentes y los aspectos dinámicos con los diagramas de interacción, estados y actividades.

7.2.7.1.15. La vista de despliegue

Un sistema contiene los nodos que forman la topología hardware sobre la que se ejecuta el sistema. Se preocupa principalmente de la distribución, entrega e instalación de las partes que constituyen el sistema. Los aspectos estáticos de esta vista se representan mediante los diagramas de despliegue y los aspectos dinámicos con diagramas de interacción, estados y actividades.

7.2.7.2. Calidad en el análisis, diseño y evaluación del software

7.2.7.2.1. Calidad de atributos

Varios atributos son generalmente considerados importantes que permiten obtener un diseño de software con alta calidad, existen algunas características que son (mantenible, portabilidad, probable) y (correctos, robusto). Cabe destacar que existen diferencias entre calidad de atributos que son (rendimiento, seguridad, funcionalidad y usabilidad), y los que son (portabilidad, reutilización, integralidad y pruebas), y las características relacionadas con la arquitectura (integridad conceptual, correcto, completo).

7.2.7.2.2. Calidad en análisis y evaluación de técnicas

Varias técnicas y herramientas pueden ayudar a mejorar la calidad de diseño de software:

7.2.7.2.2.1. Calidad en el Diseño de software

Para este tipo se puede aplicar al diseño de software informal y semi-informal tomando un grupo base, técnicas que permiten verificar la calidad de diseño de los artefactos que pueden ser (vista de la arquitectura, diseño -inspección, técnicas y requerimientos).

7.2.7.2.2.2. Análisis estático

Para este tipo se puede aplicar al diseño de software informal y semi informal que permite evaluar algo simple utilizando análisis automáticos de casos de pruebas.

7.2.7.2.2.3. Simulación y prototipos

Son técnicas dinámicas que permiten evaluar un diseño la característica de simulación, o la flexibilidad del prototipo.

7.2.7.3. Diseño de software

Muchas notaciones y lenguajes existen para representar el diseño de artefactos de software. Algunos describen un diseño estructural organizado, otros representan el inicio del software. Estas notaciones son generalmente usadas durante un diseño natural y se pueden usar durante ambos casos. Una representan notaciones que son usadas en el contexto de específicos métodos en las estrategias de diseño y métodos de sub áreas, pero estas categorías son categorizadas en notaciones para describir la estructura estática y la dinámicas vistas.

7.2.7.3.1. Software diseño estrategias y métodos

Existen varias estrategias en el desarrollo del software que permiten mejorar el diseño de procesos, a diferencia con las estrategias generales, métodos que son específicos en generar estrategias y proveen notación para ser usados en Métodos y descripción del proceso. Los métodos utilizados son medias que permiten transferir conocimiento y como un framework que permiten testear la ingeniería del software.

7.2.7.3.2. Estrategias generales

Las estrategias generales son usadas en el diseño de procesos son divididos y refinados permitiendo lograr una alta extracción de datos y información para esto utilizando heurísticas usando para esto patentes y patentes de lenguajes

7.2.8. Desarrollo de Software

7.2.8.1. Fundamentos del Análisis de Requerimientos

Es el conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software. Es la etapa más crucial del desarrollo de un proyecto de software.

La IEEE los divide en funcionales y no funcionales:

Funcionales: Condición o capacidad de un sistema requerida por el usuario para resolver un problema o alcanzar un objetivo.

No Funcionales: Condición o capacidad que debe poseer un sistema par satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.

Para realizar bien el desarrollo de software es esencial realizar una especificación completa de los requerimientos de los mismos. Independientemente de lo bien diseñado o codificado que esté, un programa pobremente especificado decepcionará al usuario y hará fracasar el desarrollo.

La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, El ámbito del programa, establecido inicialmente durante la ingeniería del sistema, es refinado en detalle. Se analizan y asignan a los distintos elementos de los programas las soluciones alternativas. Tanto el que desarrolla el software como el cliente tienen un papel activo en la especificación de requerimientos. El cliente intenta reformular su concepto, algo nebuloso, de la función y comportamiento de los programas en detalles concretos, El que desarrolla el software actúa como interrogador, consultor y el que resuelve los problemas.

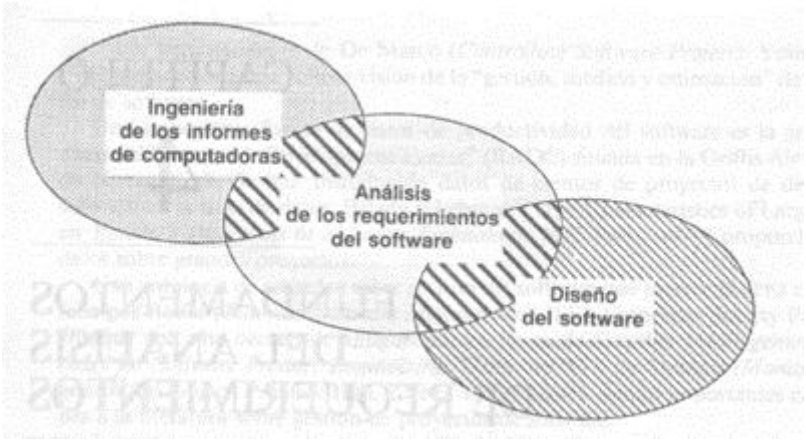
El análisis y especificación de requerimientos puede parecer una tarea relativamente sencilla, pero las apariencias engañan. Puesto que el contenido de comunicación es muy alto, abundan los cambios por mala interpretación o falta de información. El dilema con el que se enfrenta un ingeniero de software puede ser comprendido repitiendo la sentencia de un cliente anónimo: "Sé que

crees que comprendes lo que piensas que he dicho, pero no estoy seguro de que lo que creíste oír sea lo que yo quise decir".

7.2.8.2. Análisis de Requerimientos

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas. El análisis de requerimientos facilita al ingeniero de sistemas especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos del sistema y establecer las ligaduras de diseño que debe cumplir el programa. El análisis de requerimientos permite al ingeniero refinar la asignación de software y representar el dominio de la información que será tratada por el programa. El análisis de requerimientos de al diseñador la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra al técnico y al cliente, los medios para valorar la calidad de los programas, una vez que se haya construido.

Figura 4Asignación de software a nivel de sistema y el diseño de programas.



Fuente: http://www.monografias.com/trabajos5/desof/desof.shtml

7.2.8.3. Tareas del Análisis

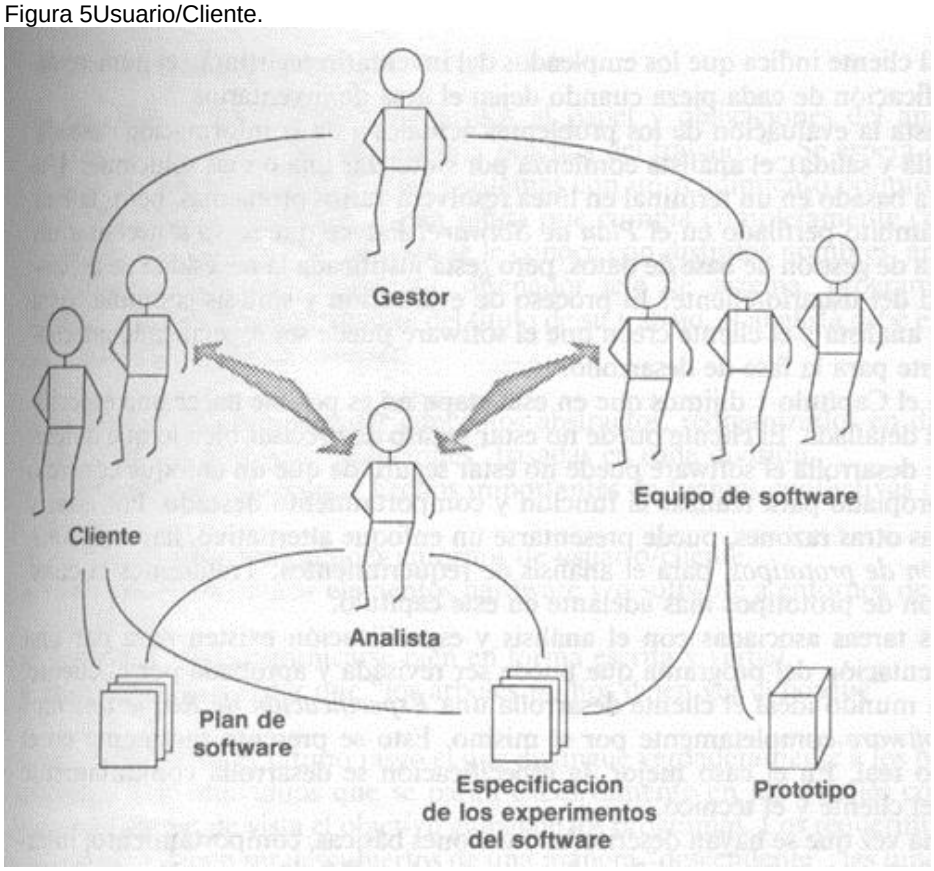
El análisis de requerimientos puede dividirse en cuatro áreas:

- Reconocimiento del problema.
- Evaluación y síntesis.
- Especificación.
- Revisión.

Inicialmente, el analista estudia la especificación del sistema (si existe) y el plan de proyecto. Es importante comprender el contexto del sistema y revisar el ámbito de los programas que se usó para generar las estimaciones de la planificación. A continuación, debe establecerse la comunicación necesaria para el análisis, de forma que se asegure el reconocimiento del problema.

El analista debe establecer contacto con el equipo técnico y de gestión del usuario/cliente y con la empresa que vaya a desarrollar el software. El gestor del programa puede servir como coordinador para facilitar el establecimiento de los caminos de comunicación. El objetivo del analista es reconocer los elementos básicos del programa tal como lo percibe el usuario/cliente.

Figura 5Usuario/Cliente.



Fuente: http://www.monografias.com/trabajos5/desof/desof.shtml

La evaluación del problema y la síntesis de la solución es la siguiente área principal de trabajo del análisis. El analista debe evaluar el flujo y estructura de la información, refinar en detalle todas las funciones del programa, establecer las características de la interface del sistema y descubrir las ligaduras del diseño, Cada una de las tareas sirve para descubrir el problema de forma que pueda sintetizarse un enfoque o solución global.

Las tareas asociadas con el análisis y especificación existen para dar una representación del programa que pueda ser revisada y aprobada por el cliente. En un mundo ideal el cliente desarrolla una especificación de requerimientos del software completamente por sí mismo. Esto se presenta raramente en el mundo real. En el mejor de los casos, la especificación se desarrolla conjuntamente entre el cliente y el técnico.

Una vez que se hayan descrito las funcionalidades básicas, comportamiento, interface e información, se especifican los criterios de validación para demostrar una comprensión de una correcta implementación de los programas. Estos criterios sirven como base para hacer una prueba durante el desarrollo de los programas. Para definir las características y atributos del software se escribe una especificación de requerimientos formal. Además, para los casos en los que se desarrolle un prototipo se realiza un manual de usuario preliminar.

Puede parecer innecesario realizar un manual de usuario en una etapa tan temprana del proceso de desarrollo, Pero de hecho, este borrador del manual de usuario fuerza al analista a tomar el punto de vista del usuario del software. El manual permite al usuario / cliente revisar el software desde una perspectiva de ingeniería humana y frecuentemente produce el comentario: "La idea es correcta pero esta no es la forma en que pensé que se podría hacer esto". Es mejor descubrir tales comentarios lo más tempranamente posible en el proceso.

Los documentos del análisis de requerimiento (especificación y manual de usuario) sirven como base para una revisión conducida por el cliente y el técnico. La revisión de los requerimientos casi siempre produce modificaciones en la función, comportamiento, representación de la información, ligaduras o criterios de validación. Además, se realiza una nueva apreciación del plan del proyecto de software para determinar si las primeras estimaciones siguen siendo válidas después del conocimiento adicional obtenido durante el análisis.

7.2.8.4. Visiones Lógicas y Físicas

La visión lógica de los requerimientos del software presenta las funciones que han de realizarse y la información que ha de procesarse independientemente de los detalles de implementación.

La visión física de los requerimientos del software presenta una manifestación del mundo real de las funciones de procesamiento y las estructuras de información. En algunos casos se desarrolla una representación física como el primer paso del diseño del software. Sin embargo la mayoría de los sistemas basados en computador, se especifican de forma que se dictan ciertas recomendaciones físicas.

7.2.8.5. Construcción de Prototipos de Software

En análisis debe ser conducido independientemente del paradigma de ingeniería de software aplicado. Sin embargo, la forma que ese análisis tomara puede variar. En algunos casos es posible aplicar los principios de análisis fundamental y derivar a una especificación en papel del software desde el cual pueda desarrollarse un diseño. En otras situaciones, se va a una recolección de los requerimientos, se aplican los principios de análisis y se construye un modelo de software, llamado un prototipo, según las apreciaciones del cliente y del que lo desarrolla. Finalmente, hay circunstancias que requieren la construcción de un prototipo al comienzo del análisis, puesto que el modelo es el único mediante el que los requerimientos pueden ser derivados efectivamente.

7.2.8.6. Un escenario para la construcción de prototipos

Todos los proyectos de ingeniería de software comienzan con una petición del cliente. La petición puede estar en la forma de una memoria que describe un problema, un informe que define un conjunto de objetivos comerciales o del producto, una petición de propuesta formal de una agencia o compañía exterior, o una especificación del sistema que ha asignado una función y comportamiento al software, como un elemento de un sistema mayor basado en computadora. Suponiendo que existe una petición para un programa de una de las formas dichas anteriormente.

7.2.8.7. Especificación

No hay duda de que la forma de especificar tiene mucho que ver con la calidad de la solución. Los ingenieros de software que se han esforzado en trabajar con especificaciones incompletas, inconsistentes o mal establecidas han experimentado la frustración y confusión que invariablemente se produce. Las consecuencias se padecen en la calidad, oportunidad y completitud del software resultante.

Hemos visto que los requerimientos de software pueden ser analizados de varias formas diferentes. Las técnicas de análisis pueden conducir a una especificación en papel que contenga las descripciones gráficas y el lenguaje natural de los requerimientos del software. La construcción de prototipos conduce a una especificación ejecutable, esto es, el prototipo sirve como una representación de los requerimientos. Los lenguajes de especificación formal conducen a representaciones formales de los requerimientos que pueden ser verificados o analizados.

7.2.9. Lenguajes de programación

Un **lenguaje de programación** es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.[14] Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos, a través de los siguientes pasos:

- El desarrollo lógico del programa para resolver un problema en particular.
- Escritura de la lógica del programa empleando un lenguaje de programación específico (codificación del programa).
- Ensamblaje o compilación del programa hasta convertirlo en lenguaje de máquina.
- Prueba y depuración del programa.
- Desarrollo de la documentación.

Existe un error común que trata por sinónimos los términos 'lenguaje de programación' y 'lenguaje informático'. Los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como por

[14](Mark) (2010). O'Reilly Media, Inc. (ed.): «Learning Python, Fourth Edition » (libro). O'Reilly. Consultado el 11 de febrero de 2010.

ejemplo HTML (lenguaje para el marcado de páginas web que no es propiamente un lenguaje de programación, sino un conjunto de instrucciones que permiten diseñar el contenido de los documentos).

Permite especificar de *manera precisa* sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar *relativamente* próximo al lenguaje humano o natural. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.

Paradigmas

Los programas se pueden clasificar por el paradigma del lenguaje que se use para producirlos. Los principales paradigmas son: imperativos, declarativos y orientación a objetos.

Los programas que usan un lenguaje imperativo especifican un algoritmo, usan declaraciones, expresiones y sentencias.[15] Una declaración asocia un nombre de variable con un tipo de dato, por ejemplo: var x: integer;. Una expresión contiene un valor, por ejemplo: 2 + 2 contiene el valor 4. Finalmente, una sentencia debe asignar una expresión a una variable o usar el valor de una variable para alterar el flujo de un programa, por ejemplo: x := 2 + 2; if x == 4 thenhaz_algo();. Una crítica común en los lenguajes imperativos es el efecto de

[15] Wilson, Leslie B. (1993). Comparative Programming Languages, Second Edition.Addison-Wesley.pp. 75. ISBN 0-201-56885-3. (eninglés).

las sentencias de asignación sobre una clase de variables llamadas "no locales".[16]

Los programas que usan un lenguaje declarativo especifican las propiedades que la salida debe conocer y no especifica cualquier detalle de implementación. Dos amplias categorías de lenguajes declarativos son los lenguajes funcionales y los lenguajes lógicos. Los lenguajes funcionales no permiten asignaciones de variables no locales, así, se hacen más fáciles, por ejemplo, programas como funciones matemáticas.[17] El principio detrás de los lenguajes lógicos es definir el problema que se quiere resolver (el objetivo) y dejar los detalles de la solución al sistema.[18] El objetivo es definido dando una lista de sub-objetivos. Cada sub-objetivo también se define dando una lista de sus sub-objetivos, etc. Si al tratar de buscar una solución, una ruta de sub-objetivos falla, entonces tal sub-objetivo se descarta y sistemáticamente se prueba otra ruta.

La forma en la cual se programa puede ser por medio de texto o de forma visual. En la programación visual los elementos son manipulados gráficamente en vez de especificarse por medio de texto.

7.2.9.1. HyperTextMarkupLanguage (HTML)

HTML, siglas de HyperTextMarkupLanguage («lenguaje de marcado de hipertexto»), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura

[16] Wilson, Leslie B. (1993). Comparative Programming Languages, Second Edition.Addison-Wesley.pp. 213. ISBN 0-201-56885-3. (eninglés).
[17] Ibid., p. 159
[18] Wilson, Leslie B. (1993). *Comparative Programming Languages, Second Edition*.Addison-Wesley.pp. 244. ISBN 0-201-56885-3. (eninglés).

y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.[19]

HTML también sirve para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

7.2.9.1.1. Accesibilidad web

El diseño en HTML, aparte de cumplir con las especificaciones propias del lenguaje, debe respetar ciertos criterios de accesibilidad web, siguiendo unas pautas o las normativas y leyes vigentes en los países donde se regule dicho concepto. Se encuentra disponible y desarrollado por el W3C a través de las Pautas de Accesibilidad al Contenido Web 1.0 WCAG (actualizadas recientemente con la especificación 2.0[20]), aunque muchos países tienen especificaciones propias, como es el caso de España con la Norma UNE 139803.[21]

[19] http://www.ri5.com.ar/ayuda03.php
[20] Web Content Accessibility Guidelines (WCAG) 2.0 - http://www.w3.org/TR/WCAG/
[21] Norma UNE 139803 - http://www.inteco.es/Accesibilidad/difusion/Normativa/Descarga/DescargaUNE_139803

7.2.9.2. HyperTextPreprocessor (PHP)

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación detallada del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica, PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por RasmusLerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

7.2.9.2.1. Definición de PHP

PHP (acrónimo de "PHP: HypertextPreprocessor") PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante sitio con otros nuevos lenguajes no tan poderosos desde agosto de 2005.

7.2.10. Apache

El **servidor HTTP Apache** es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1[22] y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, *a patchyserver*(un servidor "parcheado") suena igual que *Apache Server*.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft[23]).

[22] RFC 2616 -http://www.ietf.org/rfc/rfc2616.txt
[23] Netcraft - http://news.netcraft.com/

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

7.2.10.1. Ventajas

- Modular
- Código abierto
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

7.2.10.2. Módulos

La arquitectura del servidor Apache es muy modular. El servidor consta de una sección *core* y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web. Algunos de estos módulos son:

- mod_ssl - Comunicaciones Seguras vía TLS.
- mod_rewrite - reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- mod_dav - Soporte del protocolo WebDAV (RFC 2518).

- mod_deflate - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.
- mod_auth_ldap - Permite autentificar usuarios contra un servidor LDAP.
- mod_proxy_ajp - Conector para enlazar con el servidor JakartaTomcat de páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- mod_cband - Control de tráfico y limitador de ancho de banda.
- mod_perl - Páginas dinámicas en Perl.
- mod_php - Páginas dinámicas en PHP.
- mod_python - Páginas dinámicas en Python.
- mod_rexx - Páginas dinámicas en REXX y Object REXX.
- mod_ruby - Páginas dinámicas en Ruby.
- mod_aspdotnet - Páginas dinámicas en .NET de Microsoft **(Módulo retirado)**.
- mod_mono - Páginas dinámicas en Mono
- mod_security - Filtrado a nivel de aplicación, para seguridad.

7.2.10.3. Uso

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python (y ahora también Ruby).

Este servidor web es redistribuido como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X integra apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects. Es soportado de alguna manera por Borland en las herramientas de desarrollo Kylix y Delphi. Apache es incluido con Novell NetWare 6.5, donde es el servidor web por defecto, y en muchas distribuciones Linux.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

Microsoft Internet InformationServices (IIS) es el principal competidor de Apache, así como Sun Java System Web Server de Sun Microsystems y un anfitrión de otras aplicaciones como Zeus Web Server. Algunos de los más grandes sitios web del mundo están ejecutándose sobre Apache. La capa frontal (frontend) del motor de búsqueda Google está basada en una versión modificada de Apache, denominada Google Web Server (GWS). Muchos proyectos de Wikimedia también se ejecutan sobre servidores web Apache.

7.2.10.4. Configuración

La mayor parte de la configuración se realiza en el fichero apache2.conf o httpd.conf, según el sistema donde esté corriendo. Cualquier cambio en este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente.

7.2.10.5. Licencia

La licencia de software bajo la cual el software de la fundación Apache es distribuido es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.
La Free Software Foundation no considera a la Licencia Apache como compatible con la versión 2 de la GNU General PublicLicense (GPL), en la cual el software licenciado bajo la Apache License no puede ser integrado con software distribuido bajo la GPL:

Este es software libre pero es incompatible con la GPL. La Apache Software License es incompatible con la GPL porque tiene un requerimiento específico que no está incluido en la GPL: tiene ciertos casos de terminación de patentes que la GPL no requiere. No consideramos que dichos casos de terminación de patentes son inherentemente una mala idea, pero a pesar de ello son incompatibles con la GNU GPL.[24]

[24] Various Licenses and Comments about Them from GNU - http://www.gnu.org/philosophy/license-list.html

Sin embargo, la versión 3 de la GPL incluye una provisión (Sección 7e) que le permite ser compatible con licencias que tienen cláusulas de represalia de patentes, incluyendo a la Licencia Apache.

El nombre Apache es una marca registrada y puede ser sólo utilizada con el permiso expreso del dueño de la marca.[25]

7.2.11. Diseño de Bases de Datos

Son muchas las consideraciones a tomar en cuenta al momento de hacer el diseño de la base de datos, quizá las más fuertes sean:

- La velocidad de acceso,
- El tamaño de la información,
- El tipo de la información,
- Facilidad de acceso a la información,
- Facilidad para extraer la información requerida,
- El comportamiento del manejador de bases de datos con cada tipo de información.

No obstante que pueden desarrollarse sistemas de procesamiento de archivo e incluso manejadores de bases de datos basándose en la experiencia del equipo de desarrollo de software logrando resultados altamente aceptables, siempre es recomendable la utilización de determinados estándares de diseño que garantizan el nivel de eficiencia más alto en lo que se refiere a almacenamiento y recuperación de la información.

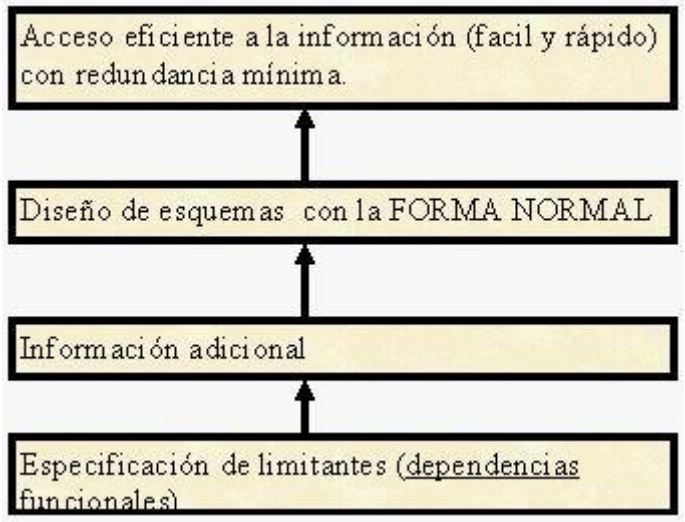
[25] Apache License and Distribution FAQ».The Apache Software Foundation (2007). - http://www.apache.org/foundation/license-faq.html#Marks

De igual manera se obtiene modelos que optimizan el aprovechamiento secundario y la sencillez y flexibilidad en las consultas que pueden proporcionarse al usuario.

7.2.11.1. Objetivos del diseño de bases de datos

Entre las metas más importantes que se persiguen al diseñar un modelo de bases de datos, se encuentran las siguientes que pueden observarse en esta figura.

Figura 6. Objetivos del diseño de bases de datos



Fuente: http://shalom-now.blogspot.com/2012/04/diseno-de-base-de-datos.html

7.2.11.2. Almacenar Solo La Información Necesaria.

A menudo pensamos en todo lo que quisiéramos que estuviera almacenado en una base de datos y diseñamos la base de datos para guardar dichos datos. Debemos de ser realistas acerca de nuestras necesidades y decidir qué información es realmente necesaria.

Frecuentemente podemos generar algunos datos sobre la marcha sin tener que almacenarlos en una tabla de una base de datos. En estos casos también tiene sentido hacer esto desde el punto de vista del desarrollo de la aplicación.

7.2.11.3. Normalizar la Estructura de las Tablas.

Si nunca antes hemos oído hablar de la "normalización de datos", no debemos temer. Mientras que la normalización puede parecer un tema complicado, nos podemos beneficiar ampliamente al entender los conceptos más elementales de la normalización.

Uno de los objetivos de una estructura de tabla normalizada es minimizar el número de "celdas vacías". El darnos cuenta de que cada lista de Cd tiene un conjunto fijo de campos (título, artista, año, género) y un conjunto variable de atributos (el número de pistas) nos da una idea de cómo dividir los datos en múltiples tablas que luego podamos relacionar entre sí.

Mucha gente no está familiarizada con el concepto "relacional", de manera sencilla esto significa, que grupos parecidos de información son almacenados en distintas tablas que luego pueden ser "juntadas" (relacionadas) basándose en los datos que tengan en común.

Es necesario que al realizar la estructura de una base de datos, esta sea flexible. La flexibilidad está en el hecho que podemos agregar datos al sistema posteriormente sin tener que rescribir lo que ya tenemos. Por ejemplo, si quisiéramos agregar la información de los artistas de cada álbum, lo único que tenemos que hacer es crear una tabla artista que esté relacionada a la tabla

álbum de la misma manera que la tabla pista. Por lo tanto, no tendremos que modificar la estructura de nuestras tablas actuales, simplemente agregar la que hace falta.

La eficiencia se refiere al hecho de que no tenemos duplicación de datos, y tampoco tenemos grandes cantidades de "celdas vacías".

El objetivo principal del diseño de bases de datos es generar tablas que modelan los registros en los que guardaremos nuestra información.

Es importante que esta información se almacene sin redundancia para que se pueda tener una recuperación rápida y eficiente de los datos.

A través de la normalización tratamos de evitar ciertos defectos que nos conduzcan a un mal diseño y que lleven a un procesamiento menos eficaz de los datos.

Podríamos decir que estos son los principales objetivos de la normalización:

- Controlar la redundancia de la información.
- Evitar pérdidas de información.
- Capacidad para representar toda la información.
- Mantener la consistencia de los datos.

7.2.11.4. Conceptos importantes

7.2.11.4.1. Base de Datos

Cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora, diseñado para facilitar su mantenimiento y acceso de una forma estándar. Los datos suelen aparecer en forma de texto, números o gráficos. Hay cuatro modelos principales de bases de datos: el modelo jerárquico, el modelo en red, el modelo relacional (el más extendido hoy en día).

7.2.11.4.2. Base de Datos Relacional

Tipo de base de datos o sistema de administración de bases de datos, que almacena información en tablas (filas y columnas de datos) y realiza búsquedas utilizando los datos de columnas especificadas de una tabla para encontrar datos adicionales en otra tabla.

7.2.11.4.3. DBMS: Data Base Management System (SISTEMA DE MANEJO DE BASE DE DATOS)

Consiste de una base de datos y un conjunto de aplicaciones (programas) para tener acceso a ellos.

7.2.11.5. MYSQL

MySQL es un sistema de gestión de bases de datos relacional, multi hilo y multiusuario con más de seis millones de instalaciones. MySQL AB—desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

7.2.12. Implementación

“La etapa de implementación del desarrollo del software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre

implica los procesos de diseño y programación de software, pero si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software."[26]

7.2.13. Pruebas

“Las pruebas del software son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. No es raro que el coste de las pruebas del software suponga un 40% del coste total de desarrollo del proyecto.
Interesa señalar que en cada fase del ciclo de vida de desarrollo del software se plantea un conjunto de pruebas que permiten constatar que el software desarrollado satisface las especificaciones de esa fase.”[27]

7.2.14. Gestión y control de la calidad

El Control de la Calidad se posesiona como una estrategia para asegurar el mejoramiento continuo de la calidad. Programa para asegurar la continua satisfacción de los clientes externos e internos mediante el desarrollo permanente de la calidad del producto y sus servicios.

Concepto que involucra la orientación de la organización a la calidad manifestada en la calidad de sus productos, servicios, desarrollo de su personal y contribución al bienestar general.

[26]Sommerville, Ian (2005), Ingeniería del software Séptima edición. Ed, Miguel Martin – Romo, España.
[27]Alonso, Fernando. Martínez, Loic. Segovia, Fco Javier (2005). Introducción a la ingeniería de software: Modelos de desarrollo de programas. Ed, Javier Barbero Rubio.

La definición de una estrategia asegura que la organización está haciendo las cosas que debe hacer para lograr sus objetivos.

La definición de su sistema determinar si está haciendo estas cosas correctamente. La calidad de los procesos se mide por el grado de adecuación de estos a lograr la satisfacción de sus clientes (internos o externos). Esto implica la definición de requerimientos del cliente o consumidor, los modos de medición y estándares contra que comparar la calidad.

8. MARCO METODOLOGICO

8.1. Tipo de investigación

La presente investigación “DESARROLLO DE UN APLICATIVO WEB QUE PERMITA MONITOREAR LOS PROYECTOS Y PRODUCTOS DE INVESTIGACIÓN QUE REALIZAN LOS DOCENTES INVESTIGADORES DE LA FACULTAD DE INGENIERÍA.” Se plantea de tipo exploratoria y propositiva, exploratoria porque al no existir investigaciones previas acerca del objeto de estudio y tener un conocimiento impreciso de este, es necesario indagar y explorar para empezar a familiarizarse con el tema, con la finalidad de alcanzar el objetivo planteado.

La investigación exploratoria finalizará cuando, a partir de los datos recolectados, se obtenga la suficiente información para saber qué temas son relevantes en la investigación y cuáles no, para desarrollar un análisis de los datos que obtuvimos y lograr dar conclusión sobre la investigación.

Propositiva porque propone el uso y aplicación de la ingeniería de software en la solución de problemas de carácter administrativo en los procesos de auditoria.

8.2. Método de investigación

Las etapas metodológicas del presente proyecto son:

- Recolección y diagnóstico de la información.
- Ingeniería de requerimientos.
- Diseño de software.
- Desarrollo de software.

8.2.1. Recolección y diagnóstico de la información

8.2.1.1. Objetivo

Obtener la mayor cantidad posible de información sobre el proceso de revisión de propuestas de investigación para ingeniería, con el fin de obtener un diagnósticopara el “DESARROLLO DE UN APLICATIVO WEB QUE PERMITA MONITOREAR LOS PROYECTOS Y PRODUCTOS DE INVESTIGACIÓN QUE REALIZAN LOS DOCENTES INVESTIGADORES DE LA FACULTAD DE INGENIERÍA.”

8.2.1.2. Plan

¿Cuáles son las unidades de análisis?
Procesos de revisión de propuesta de investigación de Ingeniería.
¿Dónde se encuentran?

En el Centro de Investigaciones de la Universidad Libre Pereira.
¿A través de que método se van a recolectar datos?
Por fuente directa de información del Centro de Investigaciones de la Universidad Libre Pereira.

8.2.1.2.1. Elementos del plan

- Variables a medir: Procesos de revisión de propuesta de investigación de Ingeniería.
- Recursos disponibles: información del Centro de Investigaciones de la Universidad Libre Pereira.

8.2.2. Ingeniería de requerimientos

8.2.2.1. Introducción

Un sistema de gestión de documentos es un sistema informático utilizado para rastrear y almacenar documentos electrónicos. Por lo general es también capaz de hacer el seguimiento de las diferentes versiones modificadas por los diferentes usuarios. El término tiene cierta superposición con los conceptos de los sistemas de gestión de contenidos. A menudo se considera como un componente de los sistemas de gestión de contenidos empresariales y en relación con la gestión de activos digitales, digitalización de documentos, sistemas de flujo de trabajo y los sistemas de gestión de registros.

A partir de finales de 1970, un número de vendedores comenzó a desarrollar sistemas de software para gestionar los documentos en papel. Estos sistemas tratan de documentos en papel, que incluye no sólo los documentos impresos y publicados, sino también fotografías, grabados, etc.

Más tarde, los desarrolladores comenzaron a escribir un segundo tipo de sistema que puede administrar los documentos electrónicos, es decir, todos aquellos documentos o archivos creados en los ordenadores, y, a menudo se almacenan en los sistemas de archivos locales de los usuarios. Los primeros sistemas de gestión de documentos electrónicos manejados ya sean los tipos de archivos de propiedad, o un número limitado de formatos de archivo.

Muchos de estos sistemas que posteriormente se conoció como los sistemas de imágenes de documentos, ya que se centró en la captura, almacenamiento, indexación y recuperación de los formatos de archivo de imagen. Sistemas de EDM evolucionaron a un punto donde los sistemas podrían administrar cualquier tipo de formato de archivo que podría almacenarse en la red.
Las aplicaciones crecieron para abarcar los documentos electrónicos, herramientas de colaboración, seguridad, flujo de trabajo y las capacidades de auditoría.

Estos sistemas permitieron una organización para capturar faxes y formularios, para guardar copias de los documentos, imágenes, y para almacenar los archivos de imagen en el repositorio para la seguridad y la recuperación rápida.

Mientras que muchos sistemas EDM almacenar documentos en su formato original, algunos sistemas de gestión de documentos basados en la web están comenzando a almacenar el contenido en forma de HTML. Estos sistemas de gestión de políticas requieren contenidos a ser importados en el sistema. Sin

embargo, una vez que el contenido se importa, el software actúa como un motor de búsqueda para que los usuarios puedan encontrar lo que buscan más rápidamente.

El formato HTML permite una mejor aplicación de las capacidades de búsqueda, como búsqueda de texto completo y frenar.[28]

8.2.2.2. Participantes del proyecto

Tabla 1 Víctor Soto Calderón (Investigador Principal)

Participante	**Víctor Soto Calderón**
Organización	Centro de Investigaciones Ingeniería Universidad Libre Pereira.
Rol	Ingeniero de Sistemas
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Autor: Víctor Soto Calderón

8.2.2.3. Objetivos del sistema

8.2.2.3.1. Objetivo general

Desarrollar un aplicativo web que permita monitorear los proyectos y productos de investigación que realizan los docentes investigadores de la facultad de ingeniería.

[28] Ibidpag 9.

8.2.2.3.2. Objetivos específicos

- Definir las variables a incluir en el aplicativo mediante un análisis a los procedimientos utilizados en el centro de investigaciones.
- Diseñar el aplicativo web aplicando ingeniería de Software.
- Desarrollar el aplicativo web utilizando modelo cliente-vista-servidor y lenguajes de programación orientados a objetos como PHP y bases de datos como MySQL.

8.2.2.4. Requisitos funcionales

Tabla 2 Validar Usuario

FRQ-0001	**Validar Usuario**
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno
Descripción	El sistema deberá *permitir al administrador y al docente validarse con un nombre de usuario (email) y una contraseña.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 3 Aceptar Usuarios Nuevos

FRQ-0002	Aceptar Usuarios Nuevos
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno
Descripción	El sistema deberá *El administrador será quien acepte el registro de usuarios nuevos para su posterior logueo.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 4 Interfaz Gráfica

FRQ-0003	Interfaz Gráfica
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno
Descripción	El sistema deberá *representar las interfaces graficas de usuario del software*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 5 Compartir Información

FRQ-0004	Compartir Información.
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno
Descripción	El sistema deberá *Permitir al administrador compartir información útil a los usuarios.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 6 Eliminar Información

FRQ-0005	Eliminar Información
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno
Descripción	El sistema deberá *Eliminar por parte del administrador la información que comparta a los usuarios.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 7 Agregar Proyectos de Investigación

FRQ-0006	Agregar Proyectos de Investigación
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *Agregar un nuevo proyecto de investigación por parte de un usuario (docente y/o administrador).*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 8 Eliminar Proyectos de Investigación

FRQ-0007	Eliminar Proyectos de investigación
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir eliminar proyectos de investigación propios del usuario que este logueado en el momento antes de que el administrador suba la propuesta aprobada.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado

Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 9 Aprobar propuesta de Proyecto de Investigación

FRQ-0008	**Aprobar Propuesta de Proyecto de Investigación**
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Centro de Investigaciones Ingeniería. Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir subir una propuesta aprobada por parte del administrador.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 10 Subir Informes a los proyectos de Investigación

FRQ-0009	**Subir Informes a los Proyectos de Investigación**
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir subir informes (archivos) para cada proyecto de investigación existente.*

Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 11 Ver informes de los Proyectos de Investigación

FRQ-0010	**Ver informes de los Proyectos de Investigación**
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir ver y descargar los informes que se hayan subido para cada proyecto de investigación existente.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 12 Ver Reporte de Usuarios Sin subir Informes

FRQ-0011	**Ver Reporte de Usuarios Sin subir Informes**
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Usuario

Dependencias	Ninguno
Descripción	El sistema deberá *permitir al administrador ver quienes están en mora con los informes de sus proyectos de investigación.*
Importancia	quedaría bien
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 13 Subir Documento -Final

FRQ-0012	**Subir Documento -Final**
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir subir el documento final (archivo .pdfunicamente) para el proyecto de investigación.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 14 Ver Documento Final

FRQ-0013	Ver Documento Final
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir ver o descargar el documento final de los proyectos de investigación.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 15 Enviar solicitudes al administrador

FRQ-0014	Enviar solicitudes al administrador
Versión	1.0 (02/10/2013)
Autores	Usuario
Fuentes	Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir enviar solicitudes por parte de los usuarios al administrador.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

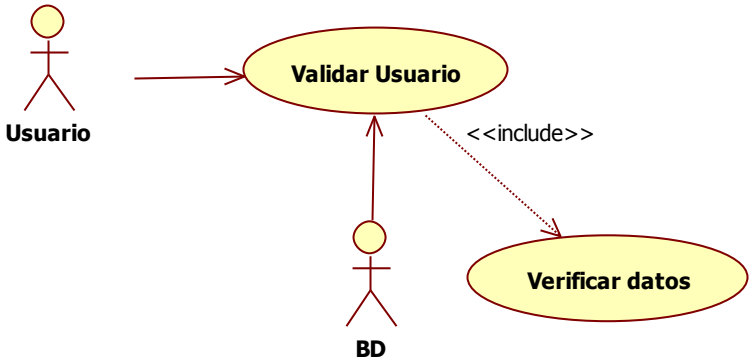
Tabla 16 Ver y Gestionar Solicitudes

FRQ-0015	Ver y Gestionar Solicitudes
Versión	1.0 (02/10/2013)
Autores	Admin
Fuentes	Usuario
Dependencias	Ninguno
Descripción	El sistema deberá *permitir ver las solicitudes enviadas por los usuariospara ser gestionadas posteriormente.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

8.2.2.5. Diagramas y documentación de casos de uso

Figura 7Diagrama caso de uso - Validar usuario



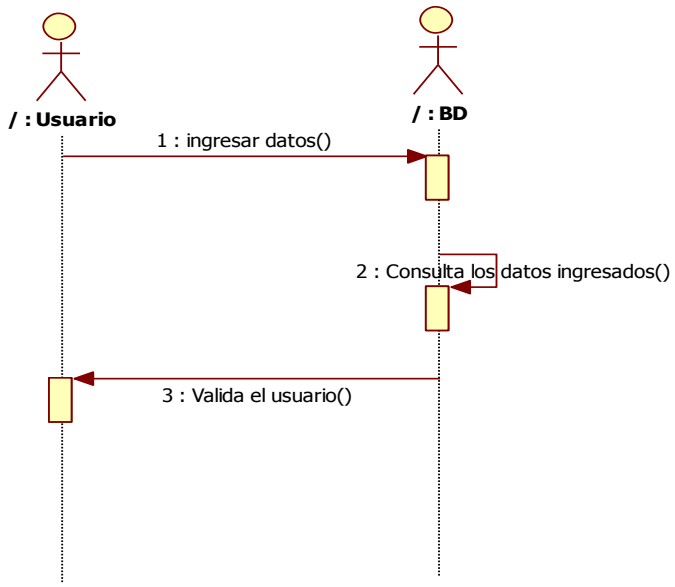
Autor: Víctor Soto Calderón

Tabla 17Documentación caso de uso Validar Usuario

UC-0001	**Validar Usuario**	
Versión	1.0 (02/10/2013)	
Autores	Admin Usuario	
Fuentes	**?**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Diligenciar formulario de registro.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema *compara los datos introducidos por el usuario con los de la BD.*
	2	El sistema *Verifica nombre de usuario y contraseña*
Postcondición	Nombre de usuario y contraseña correctos.	
Excepciones	**Paso**	**Acción**
	2	Si *Nombre de usuario y contraseña son erroneos.*, el sistema*redirecciona al usuario a la pagina principal.*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

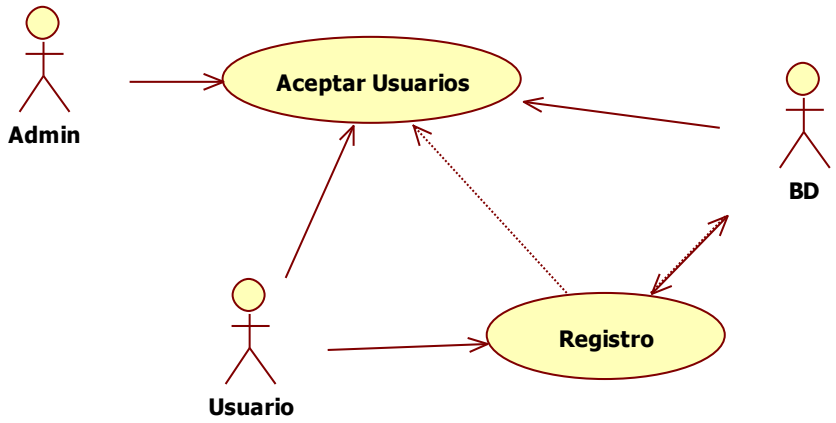
Autor: Víctor Soto Calderón

Figura 8Diagrama de secuencia – Validar Usuario



Autor: Víctor Soto Calderón

Figura 9Diagrama caso de uso – Aceptar Usuarios Nuevos



Autor: Víctor Soto Calderón

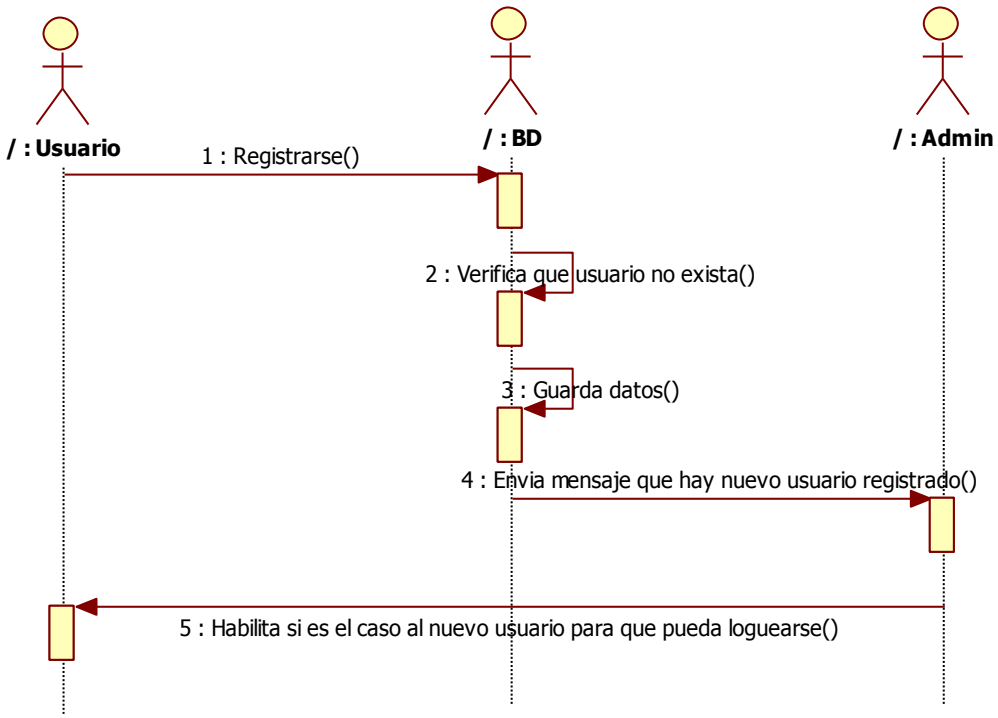
Tabla 18Documentación caso de uso – Aceptar Usuarios Nuevos

UC-0002	Aceptar Usuarios Nuevos
Versión	1.0 (02/10/2013)
Autores	Admin

Fuentes	**?**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	haber dilgenciado el formulario de registro	
Secuencia normal	**Paso**	**Acción**
	1	El actorAdmin (ACT-0001) *Acepta el nuevo usuario si lo ve pertinente.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si *Docente no pertenece a la facultad de ingeniería.*, el actorAdmin (ACT-0001) *Rechazara su petición.*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Autor: Víctor Soto Calderón

Figura 10Diagrama de secuencia – Aceptar Usuarios Nuevos



Autor: Víctor Soto Calderón

Figura 11Diagrama caso de uso – Interfaz Gráfica

Autor: Víctor Soto Calderón

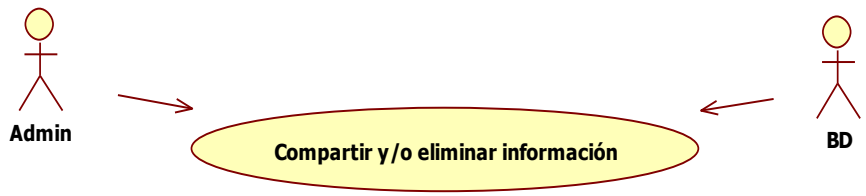
Tabla 19Documentación caso de uso – Interfaz Gráfica

UC-0003	**Interfaz Gráfica**
Versión	1.0 (02/10/2013)
Autores	Centro de Investigaciones Ingeniería.
Fuentes	?
Dependencias	Ninguno

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	tener acceso a internet el equipo de trabajo	
Secuencia normal	**Paso**	**Acción**
	-	-
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Autor: Víctor Soto Calderón

Figura 12Diagrama caso de uso – Compartir y/o Eliminar Información



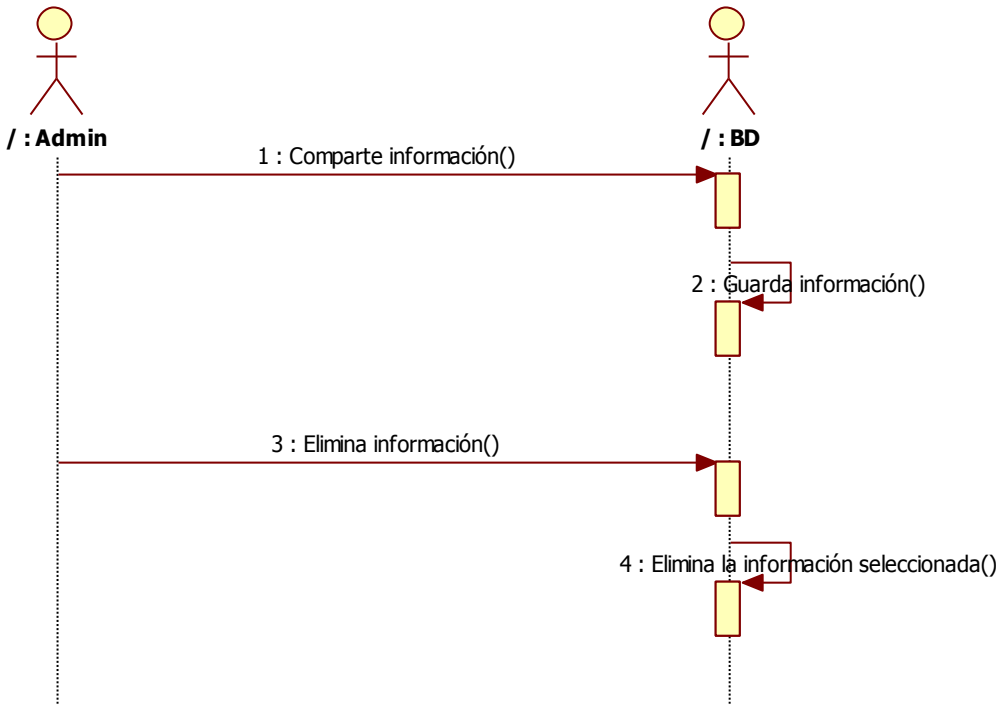
Autor: Víctor Soto Calderón

Tabla 20Documentación caso de uso – Compartir y Eliminar Información

UC-0004	**Compartir y eliminar Información**	
Versión	1.0 (02/10/2013)	
Autores	Admin	
Fuentes	Centro de Investigaciones Ingeniería.	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Estar logueado	
Secuencia normal	**Paso**	**Acción**
	1	El actorAdmin (ACT-0001) *Es el único que podra publicar y/o eliminar información.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si *No esta logueado como administrador,* el sistema *No permitira publicar y/o eliminar información.*, a continuación este caso de uso *continúa*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

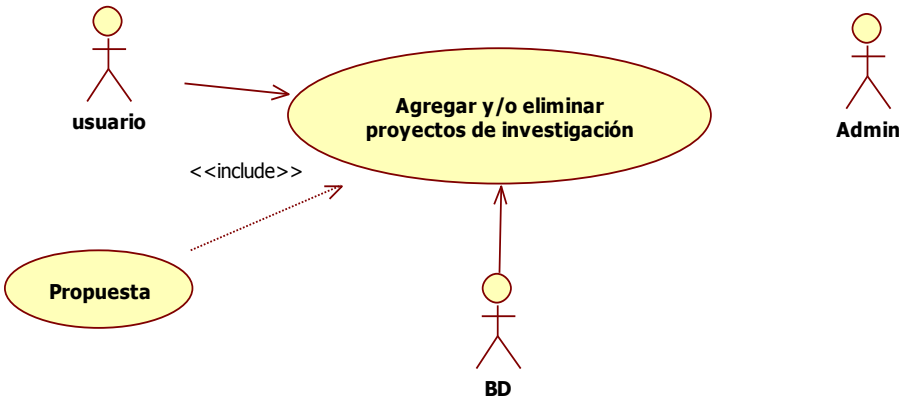
Autor: Víctor Soto Calderón

Figura 13Diagrama de secuencia – Compartir y/o Eliminar información



Autor: Víctor Soto Calderón

Figura 14Diagrama caso de uso – Agregar y/o eliminar Proyectos de Investigación



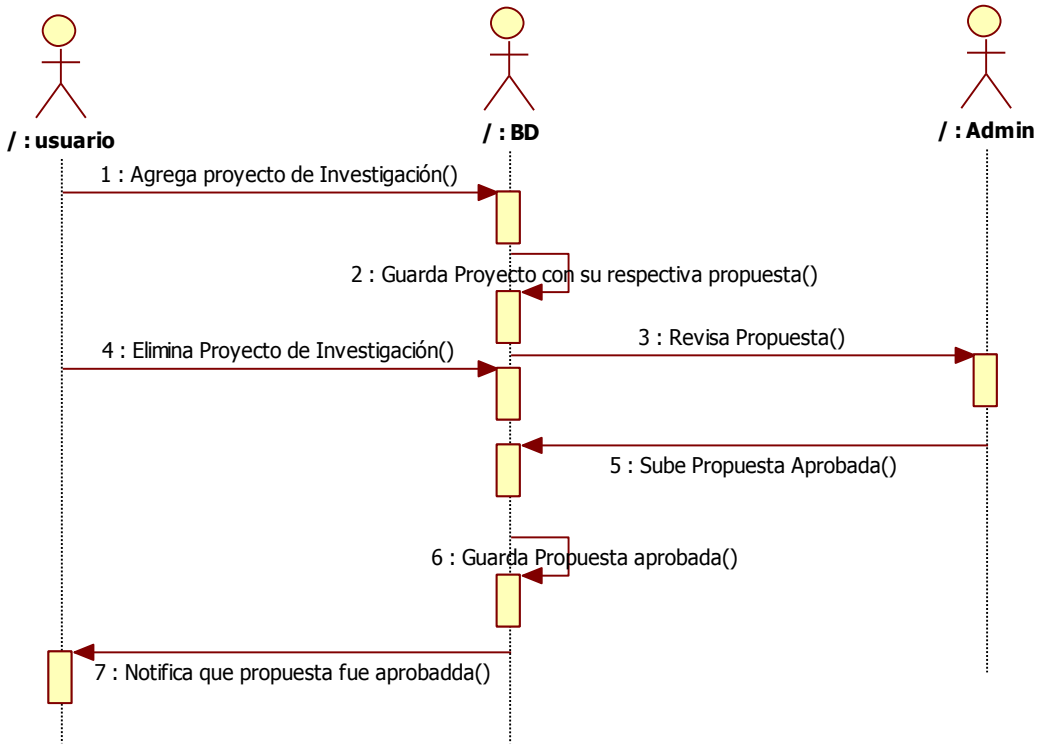
Autor: Víctor Soto Calderón

Tabla 21Documentación caso de uso – Agregar y/o Eliminar Proyectos de Investigación

UC-0005	**Agregar y/o eliminar Proyectos de investigación**	
Versión	1.0 (02/10/2013)	
Autores	Admin Usuario	
Fuentes	Admin Usuario	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Tener una propuesta redactada en un archivo (.doc, docx, .xls, xlsx, o .pdf.)	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0002) *(El administrador también hace papel de usuario para esta funcionalidad) Podra agregar un proyecto de investigación.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si, el actor Usuario (ACT-0002) *podrá eliminar proyectos de investigación propios antes que el administrador suba la apropuesta aprobada.*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

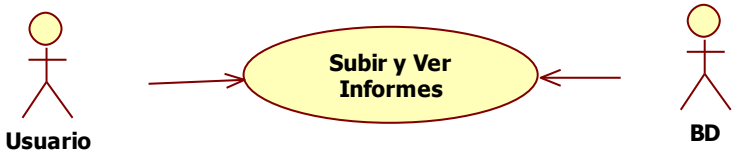
Autor: Víctor Soto Calderón

Figura 15Diagrama de secuencia – Agregar y/o Eliminar proyecto de Investigación



Autor: Víctor Soto Calderón

Figura 16Diagrama caso de uso – Subir y Ver Informes



Autor: Víctor Soto Calderón

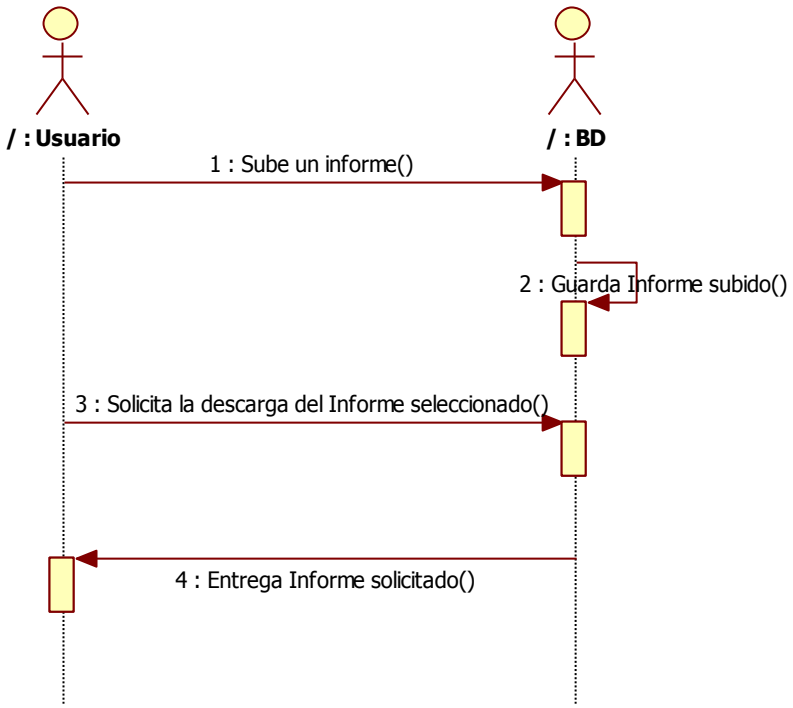
Tabla 22Documentación caso de uso – Subir y Ver Informes de los Proyectos de Investigación

UC-0006	**Subir y Ver Informes de los Proyectos de Investigación**
Versión	1.0 (02/10/2013)
Autores	Admin Usuario
Fuentes	Admin

	Usuario	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Tener un proyecto de investigación agregado.	
Secuencia normal	**Paso**	**Acción**
	1	Si *Tiene un proyecto de investigación agregado. (El administrador también hace papel de usuario para esta funcionalidad),* el actor Usuario (ACT-0002) *podrá subir los informes correspondientes a los proyectos de investigación propios.*
	2	Si *subió informes.,* el actor Usuario (ACT-0002) *podrá ver y/o descargar los informes.*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Autor: Víctor Soto Calderón

Figura 17Diagrama de secuencia – Subir y Ver Informes



Autor: Víctor Soto Calderón

Figura 18Diagrama caso de uso – Ver Reporte de Usuarios sin Subir Informes

Autor: Víctor Soto Calderón

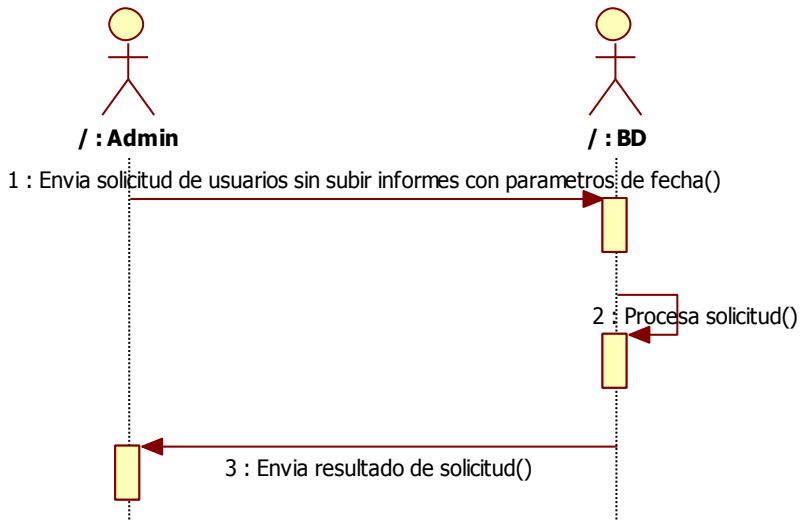
Tabla 23Documentación caso de uso – Ver Reporte de Usuarios sin Subir Informes

UC-0007	**Ver Reporte de Usuarios sin Subir Informes**
Versión	1.0 (03/10/2013)
Autores	Admin
Fuentes	Centro de Investigaciones Ingeniería.
Dependencias	Ninguno

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Estar logueado como administrador	
Secuencia normal	**Paso**	**Acción**
	1	El actorAdmin (ACT-0001) *puede ver el reporte de usuarios morosos de informes.*
Postcondición	**PD**	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

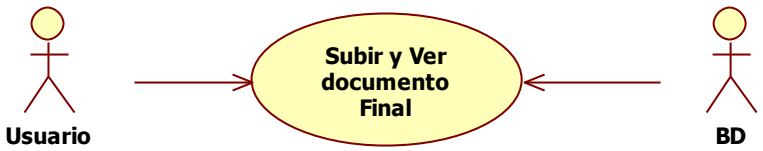
Autor: Víctor Soto Calderón

Figura 19Diagrama de secuencia – Ver Reporte de Usuarios sin Subir Informes



Autor: Víctor Soto Calderón

Figura 20 Subir y Ver Documento Final



Autor: Víctor Soto Calderón

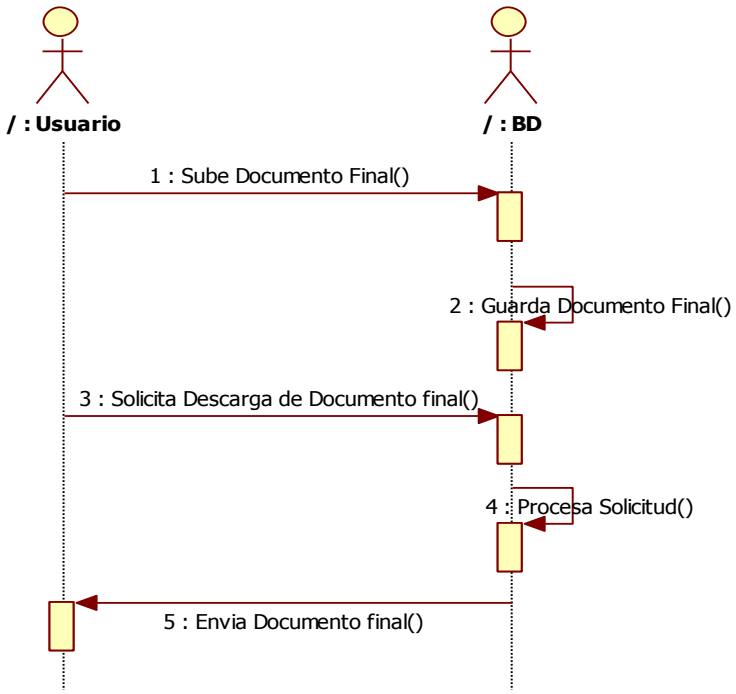
Tabla 24Documentación caso de uso – Subir y Ver documento Final

UC-0008	**Subir y Ver Documento Final**	
Versión	1.0 (03/10/2013)	
Autores	Admin Usuario	
Fuentes	Admin Usuario	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Tener propuesta aprobada y la cantidad de informes necesarios.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0002) *tiene un proyecto de investigación agregado. (El administrador también hace papel de usuario para esta funcionalidad), podrá subir el documento final correspondiente a los proyectos de investigación propios.*
Postcondición	**PD**	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	

Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Figura 21Diagrama de secuencia – Subir y Ver Documento Final



Autor: Víctor Soto Calderón

Figura 22Diagrama caso de uso – Enviar Solicitudes

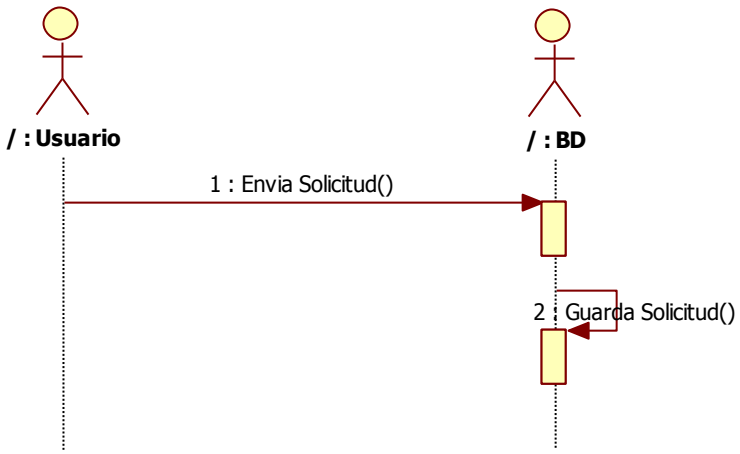
Autor: Víctor Soto Calderón

Tabla 25Documentación caso de uso – Enviar Solicitudes

UC-0009	**Enviar Solicitudes**	
Versión	1.0 (03/10/2013)	
Autores	Usuario	
Fuentes	Usuario	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Estar logueado como Usuario.	
Secuencia normal	**Paso**	**Acción**
	-	-
Postcondición	**PD**	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Autor: Víctor Soto Calderón

Figura 23Diagrama de secuencia – Enviar Solicitudes



Autor: Víctor Soto Calderón

Figura 24Diagrama caso de uso – Ver y Gestionar Solicitudes

Autor: Víctor Soto Calderón

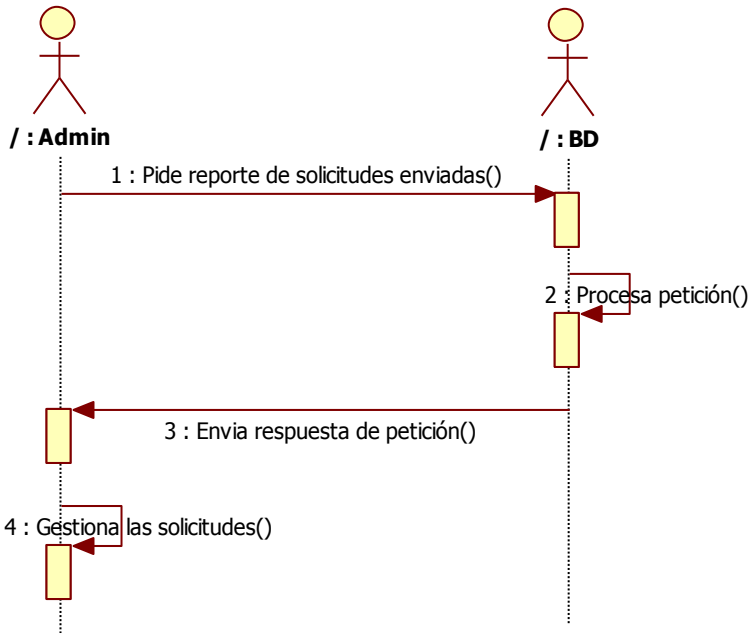
Tabla 26Documentación caso de uso – Ver y Gestionar Solicitudes

UC-0010	**Ver y Gestionar Solicitudes**	
Versión	1.0 (03/10/2013)	
Autores	**?**	
Fuentes	**?**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando	
Precondición	Estar logueado como administrador	
Secuencia normal	**Paso**	**Acción**
	1	El actorAdmin (ACT-0001) *revisa el buzón de solicitudes.*

	2	El actorAdmin (ACT-0001) *envia respuesta a quien le hizo la solicitud.*
Postcondición	**PD**	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	**PD**	
Importancia	importante	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Autor: Víctor Soto Calderón

Figura 25Diagrama de secuencia – Ver y Gestionar Solicitudes



Autor: Víctor Soto Calderón

8.2.2.6. Requisitos no funcionales

Tabla 27 Ingeniería de software

NFR-0001	**Ingeniería de software**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *tener la disciplina o área de la informática o ciencia de la computación, que ofrece conocimientos, técnicas y métodos para desarrollar y mantener software de calidad que resuelva problemas de todo tipo.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 28 Ingeniería de requerimientos

NFR-0002	**Ingeniería de requerimientos**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *comprender todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para el desarrollo de un software, tomando en cuenta los diversos requisitos a satisfacer en el desarrollo de un aplicativo web que permita monitorear los proyectos y productos de investigación que realizan los docentes investigadores de la facultad de ingeniería.*

Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 29RationalUnifedProcess (RUP)

NFR-0003	**RationalUnifedProcess (RUP)**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *contener el proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, que para este proyecto constituye la metodología estándar que se utilizara para el análisis, diseño, desarrollo, implementación y documentación delaplicativo web.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	media
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 30 Elicitación

NFR-0004	**Elicitación**
Autores	Víctor Soto Calderón
Dependencias	Ninguno

Descripción	El sistema deberá *tener en la presente investigación es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos del problema formulado.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 31Requirements Management (REM)

NFR-0005	**Requirements Management (REM)**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá contener una herramienta de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos del proyecto, se encuentra basada en la metodología definida en la Tesis Doctoral "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información", presentada por Amador Durán en septiembre de 2000.
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	media
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 32 UML

NFR-0006	UML
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *usar un lenguaje para especificar, construir, visualizar y documentar los artefactos del sistema de información a desarrollar, en la presente investigación un artefacto es información que es utilizada o producida mediante un proceso de desarrollo de software.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 33 Diseño de software

NFR-0007	Diseño de Software
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física. (E. Taylor)*
Importancia	vital
Urgencia	inmediatamente
Estado	pendiente de verificación
Estabilidad	alta

Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 34 Desarrollo de software

NFR-0008	**Desarrollo de Software**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *las metodologías de desarrollo y programación, tal como el desarrollo orientado a objetos, modelos cliente servidor, arquitecturas de bases de datos y de redes y lenguajes hipertextuales.*
Importancia	vital
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 35 Lenguajes de Programación

NFR-0009	**Lenguajes de programación**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *hacer uso de lenguajes de programación, los cuales son idiomas artificiales diseñados para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras, pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de*

	comunicación humana, está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. *Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa o software se le llama programación.* *Desarrollo Orientado a Objetos.* *Se utilizara como metodología de programación cuyo soporte fundamental será el objeto. Es un modo de trabajo más natural, que permite al desarrollador centrarse en solucionar el problema en lugar de tener que estar pensando en cómo decirle a la computadora que haga esto o lo otro.*
Importancia	importante
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 36HyperTextMarkupLanguage (HTML)

NFR-0010	**HyperTextMarkupLanguage (HTML)**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *usar el lenguaje de etiquetas predominante para la elaboración de páginas web para describir la estructura y el contenido en forma de texto, así como para complementar*

	el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento. *Este lenguaje de etiquetas se utiliza en el presente proyecto para la visualización y gestión de los datos en el modelo cliente/servidor.*
Importancia	importante
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 37HypeTextPreprocessor (PHP)

NFR-0012	**HyperTextPreprocessor (PHP)**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *usar el lenguaje de código abierto especialmente adecuado para desarrollo web y que puede ser incrustado en HTML, diseñado originalmente para la creación de páginas web dinámicas. Es el lenguaje de programación utilizado para el desarrollo de esta investigación.* *PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (y de ese tipo, como Linux o Mac OS X) y Microsoft Windows, y puede interactuar con los servidores de web más populares*

	ya que existe en versión CGI, módulo para Apache, e ISAPI. *Permite la conexión a todo tipo de servidores de base de datos como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.*
Importancia	importante
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 38 Bases de datos

NFR-0014	**Bases de datos**
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá almacenar todos los datos, archivos e imágenes de cada usuario.
Importancia	vital
Urgencia	inmediatamente
Estado	en construcción
Estabilidad	media
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 39 MySQL

NFR-0015	MYSQL
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *utilizar el software MySQL que proporcionara un servidor de base de datos SQL (StructuredQueryLanguage) veloz, multi-hilo, multiusuario y robusto. El servidor está proyectado tanto para sistemas críticos en producción soportando intensas cargas de trabajo como para empotrarse en sistemas de desarrollo masivo de software. El software MySQL tiene licencia dual, pudiéndose usar de forma gratuita bajo licencia GNU o bien adquiriendo licencias comerciales de MySQL AB en el caso de no desear estar sujeto a los términos de la licencia GPL. MySQL es una marca registrada de MySQL AB.*
Importancia	importante
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

Tabla 40 Gestión y control de calidad

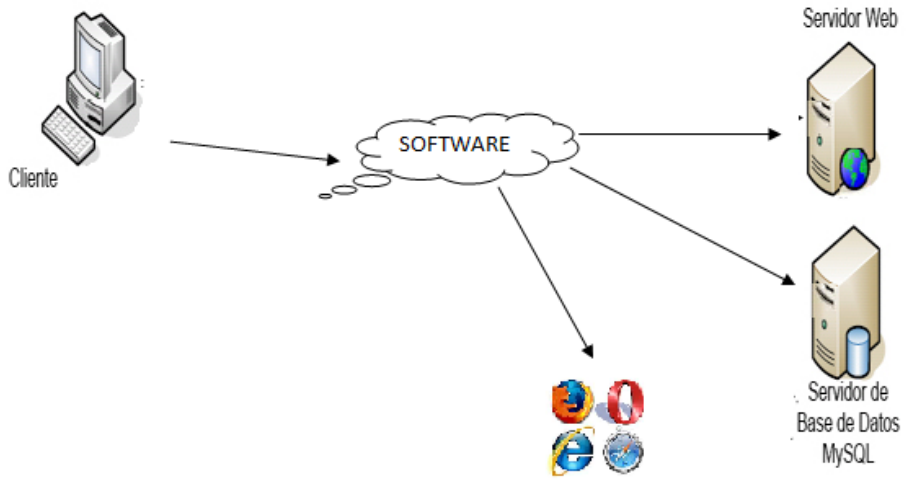
NFR-0019	Gestión y control de la calidad
Autores	Víctor Soto Calderón
Dependencias	Ninguno
Descripción	El sistema deberá *utilizar las técnicas y actividades de carácter operativo para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:*

	**Mantener bajo control un proceso.* ** eliminar las causas de los defectos en las diferentes fases del ciclo de vida.*
Importancia	importante
Urgencia	hay presión
Estado	en construcción
Estabilidad	baja
Comentarios	Ninguno

Autor: Víctor Soto Calderón

8.2.2.7. Diseño de software

Figura 26Diseño Arquitectónico



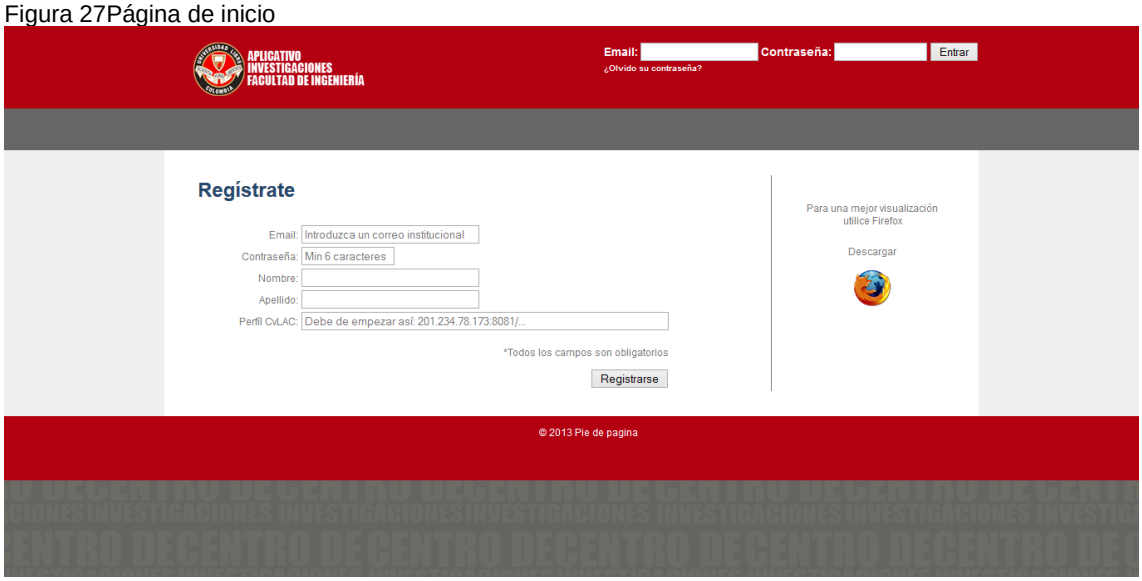
Autor: Víctor Soto Calderón

8.2.2.7.1. Interfaces graficas de usuario

8.2.2.7.1.1. Interfaz de inicio

En esta pantalla se muestra un formulario de registro y un formulario de logueo para el usuario, en el momento en que entra al sitio:

Figura 27Página de inicio

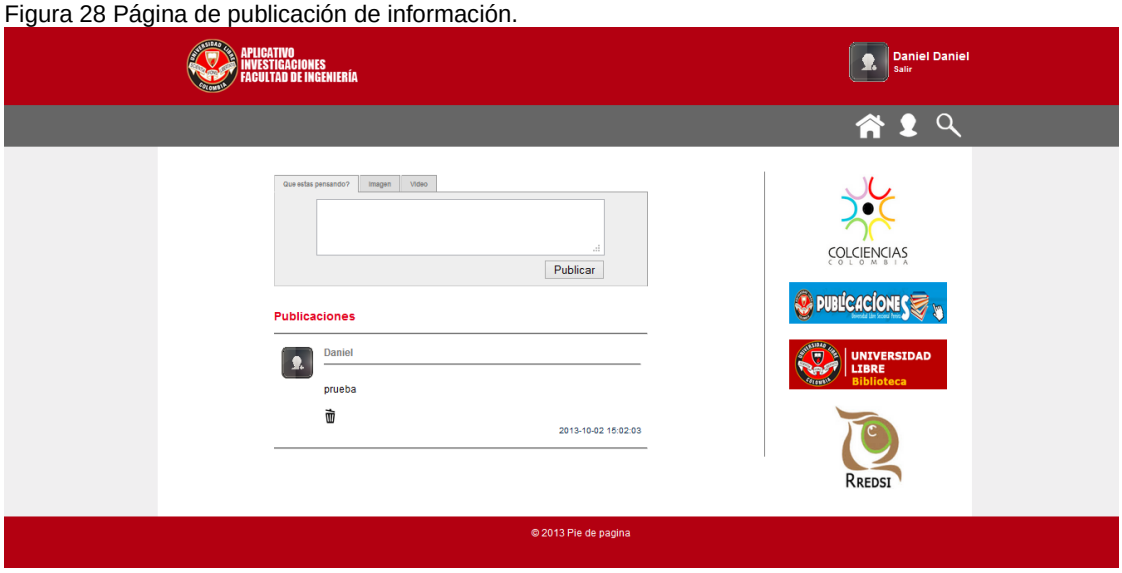


Autor: Víctor Soto Calderón

8.2.2.7.1.2. Interfaz página de publicaciones

Es el muro donde el administrador podra publicar información relevante para los usuarios del aplicativo, tiene enlaces importantes a páginas de gran interes para los usuarios, como también para ir al perfil, y a las investigaciones.

Figura 28 Página de publicación de información.

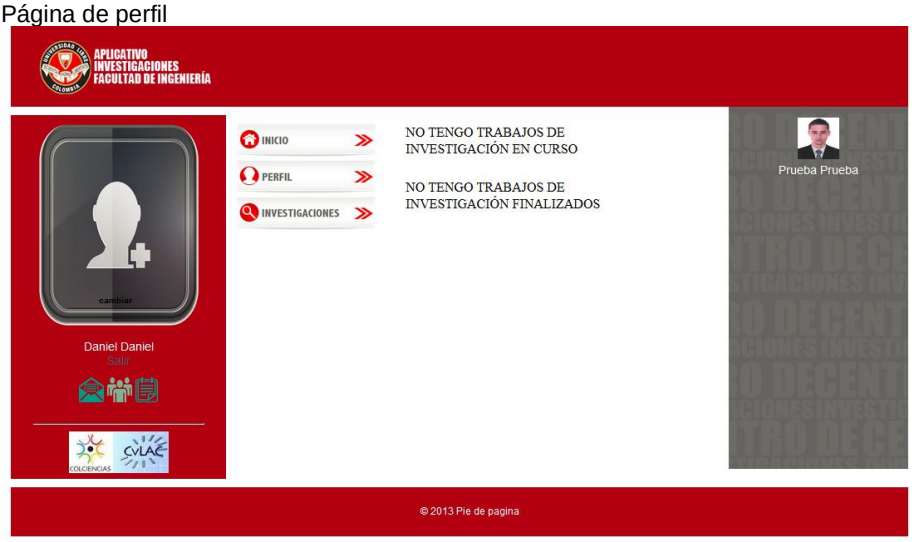


Autor: Víctor Soto Calderón

8.2.2.7.1.3. Interfaz de perfil de usuario

Tiene información básica del usuario como sus investigaciones en curso y sus investigaciones finalizadas, enlace al perfil de Colciencias y vista de otros usuarios registrados en el aplicativo.

Figura 29 Página de perfil



Autor: Víctor Soto Calderón

8.2.2.7.1.4. Interfaz página de proyectos de investigación

Aquí el usuario agregara un proyecto de investigación e irá viendo el avance de dicho proyecto, tiene enlaces para descargar los archivos subidos a determinado proyecto, para la página de perfil y la de publicaciones.

Figura 30 Página de proyectos de investigación

Autor: Víctor Soto Calderón

8.2.2.8. Desarrollo

8.2.2.8.1. Manual de instalación WAMP Server

https://www.youtube.com/watch?v=7LfVHhmyj4M

8.3. MARCO LEGAL Y NORMATIVO

Lineamientos para la implementación deprocesos electrónicos
http://programa.gobiernoenlinea.gov.co/apc-aa-files/da4567033d075590cd3050598756222c/ProcedimientosAdministrativosElectronicos_MarcoNormativo.pdf

Ley de TIC de Colombia (L1341/09)
http://www.cepal.org/publicaciones/xml/1/43371/LC-BOG-L.22.pdf

Normas procesales con uso de las TIC:
http://www.cej.org.co/index.php/marco-normativo

9. RECURSOS DISPONIBLES

RUBROS		
PERSONAL		10.000.000
EQUIPOS	COMPRA	3.000.000
	ARRIENDO	
MATERIALES E INSUMOS		500.000
SERVICIOS TÉCNICOS		500.000
CAPACITACION		5.000.000
VIAJES		0
SOFTWARE		2.000.000
MANTENIMIENTO		0
ADECUACION DE INFRAESTRUCTURA		0
GASTOS DE REGISTRO DE PROPIEDAD INTELECTUAL (sólo en el caso de patentes)		2.000.000
BIBLIOGRAFÍA	SUSCRIPCIONES	0
	LIBROS	400.000
OTROS (DISCRIMINAR)		0
TOTAL		**23.400.000**

10. CRONOGRAMA

Tabla 41 Cronograma de actividades

Año 2013

N°	Actividad	Tiempo	Control	Mes 1				Mes 2				Mes 3			
1	Idea del proyecto - diagnostico	3	E												
			R												
2	Diseño del aplicativo	10	E												
			R												
3	Desarrollo del aplicativo	200	E												
			R												
4	Implementación del aplicativo	48	E												
			R												

E = Tiempo estimado en semanas
R = Tiempo real en semanas

Autor: Víctor Soto Calderón

11. BIBLIOGRAFIA

- AGUILA SÁNCHEZ, Luis. ¨Control de la Calidad¨ Editorial Minerva, 1997 Pág. 45-50

- ANASTASI, Maribel. uml;Control de Calidad¨. Editorial AGUILAR. 1992. Lima. Pág. 75-78

- ¿Qué le falta a UML? un artículo por Scott Ambler, 'ObjetMagacine', Octubre de 1997, SIGS

- BAMNET, Jeanne. ¨Control de la Calidad¨- Editorial Fontanella. Barcelona España 1991. Pág. 45-46

- Balzer, R., and N. Goodman, "Principles of Good Software Specification", Proc, on Specifications of Reliable Software, IEEE, 1979, pp.58-67.

- Berry, J.K. (1993) *Beyond Mapping: Concepts, Algorithms and Issues in GIS*. Fort Collins, CO: GIS World Books.

- Los Mejores Trucos (Anaya Multimedia): Bluttman, Ken.

- Bolstad, P. (2005) *GIS Fundamentals: A first text on Geographic Information Systems, Second Edition*. White Bear Lake, MN: Eider Press, 543 pp.

- Bosque Sendra, J. (1992) *Sistemas de Información Geográfica*. Rialp. Madrid.

- (Booch 1998)Booch G. 1998. Software Architecture and the UML. Presentación disponible en: http://www.rational.com/uml como arch.zip.(Booch 1986)Booch, G. 1988. Object Oriented Development. Trans.

of Soft. Eng. Vol. SE-12. Num. 2. Feb. 1986. pp. 211-221.(Conallen 1999A) Conallen, J. "Modeling Web Applications with UML" Conallen, Inc. 9-Mar-1999 Disponible en:http://www.conallen.com/whitepapers/webapps/ModelingWebApplications.htm

- BRYANT J. Cartty. ¨Control de Calidad¨. Editorial Pax, México 1998. Pág. 75

- Burrough, P.A. and McDonnell, R.A. (1998) *Principles of geographical information systems*. Oxford UniversityPress, Oxford, 327 pp.

- Buzai, G.D.; Baxendale, C.A. (2006) *Análisis Socioespacial con Sistemas de Información Geográfica*. Buenos Aires, Lugar Editorial, 400 pp.

- Luis Miguel Cabezas Granado Manual Imprescindible de PHP5 Edit. ANAYAMadrid 2004

- Calvo, M. (1992) *Sistemas de Información Geográfica Digitales: Sistemas geomáticos*. IVAP-EUSKOIKER, Oñati, 616 pp.

- Calvo, M. (2012) "Geo-conceptualización y modelado del espacio geográfico". EAE. Saarbrücken, 492 pp.

- CASTELLANO, Maria. ¨Calidad total. Editorial La prensa Medica. México 1998
- Chang, K. (2007) *Introduction to Geographic Information System, 4th Edition*. McGraw Hill.

- Cochran, W. G. (1986). Técnicas de muestreo. México: Continental

- (Conallen 1999B)Conallen, J. "UML Extension for Web Applications 0.91" Drafted Conallen, Inc. 22-Mar-1999 Disponible en:http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm
- (Cota 1994)Cota A. 1994 "Ingeniería de Software". Soluciones Avanzadas. Julio de 1994. pp. 5-13.(Greiff 1994)Greiff W. R. Paradigma vs Metodología; El Caso de la POO (Parte II).*Soluciones Avanzadas*. Ene-Feb 1994. pp. 31-39.(Jacobson 1998)Jacobson, I. 1998. "Applying UML in The Unified Process" Presentación. Rational Software. Presentación disponible en http://www.rational.com/uml como UMLconf.zip

- Coulman, Ross (2001 - present) Numerous GIS White Papers

- Cramer, H. (1977). Elementos de la teoría de probabilidades. Madrid: Aguilar

- Objetos, componentes y Estructuras con UML, TheCatalysisAproach, por Desmond F. D'Souza yAlan C. Wills, Addison Wesley Longman, 1998.

- Elangovan, K (2006)"GIS: Fundamentals, Applications and Implementations", New India Publishing Agency, New Delhi"208 pp.

- Freeman, P., "Requirements Analysis and Specification", Proc. Intl. Computer Technology Conf., ASME, San Francisco, August, 1980

- Entendiendo UML: La guía del desarrollador, con una aplicación java basada en web, por PaulHarmon y Mark Watson; Morgan KauffmanPublishers, Inc., 1998
- (www.mkp.com/books_catalog/1-55860-465-0.asp).

- Harvey, Francis (2008) *A Primer of GIS, Fundamental geographic and cartographic concepts.* The Guilford Press, 31 pp.

- Heywood, I., Cornelius, S., and Carver, S. (2006) *An Introduction to Geographical Information Systems*. Prentice Hall. 3rd edition.

- (Jacobson 1992)Jacobson, I. *et*. *al*. 1992. Object-Oriented Software Engineering; A Use Case Driven Aproach. ACM Press. Adison-Wesley Publishing. Co. U.S.A. 524 p. Ilus. pp. 465-493.(Lewis 1994)Lewis G. 1994. "What is Software Engineering?" DataPro (4015). Feb 1994. pp. 1-10.(Microsoft 1997)Microsoft 1997. Microsoft Solutions Framework 1.0. Microsoft Corporation. USA.(M&R 1998)Microsoft y Rational 1998. *A White Paper on the Benefits of Integrating Microsoft Solutions Framework and The Rational Process.* Rational Software Corporation y Microsoft Corporation. Documento msfratprocs.doc Disponible en http://www.rational.com/uml/papers.

- Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W. (2005) *Geographic Information Systems and Science*. Chichester: Wiley. 2nd edition.
- López Cachero, M. (1996). Fundamentos y Métodos de la Estadística. Madrid: Ed Pirámide Quesada, V.; Isidoro, A. y López, L.A. (1989). Curso y ejercicios de estadística. Madrid: Alhambra Ríos, S. (1983). Análisis estadístico aplicado. Madrid: Paraninfo
- Maguire, D.J., Goodchild M.F., Rhind D.W. (1997) "Geographic Information Systems: principles, and applications" Longman Scientific and Technical, Harlow.
- Olaya, V., (2012) *Sistemas de Información Geográfica*. VíctorOlaya. 877 pp. (Creative Common Atribución).

- Ott, T. and Swiaczny, F. (2001) *Time-integrative GIS. Management and analysis of spatio-temporal data*, Berlin / Heidelberg / New York: Springer.

- Ríos, S. (1994). Iniciación a la estadística. Madrid: Paraninfo

- PRESSMAN Roger S., 2005. Ingeniería de Software. Un enfoque práctico. Sexta edición. 2005, Estados Unidos*.*

- Roger S. Pressman, "Ingeniería del Software: Un enfoque práctico", Segunda edición, Editorial McGraw Hill, 1990

- Introducción a las Bases de Datos**:** THOMSON PARANINFO, S.A. 2005

- Thurston, J., Poiker, T.K. and J. Patrick Moore. (2003) *Integrated Geospatial Technologies: A Guide to GPS, GIS, and Data Logging*. Hoboken, New Jersey: Wiley.

- Tomlin, C.Dana (1991) *Geographic Information Systems and Cartographic Modelling*. Prentice Hall. New Jersey.

- Tomlinson, R.F., (2005) *Thinking About GIS: Geographic Information System Planning for Managers*. ESRI Press. 328 pp.

- Smith M J, Goodchild M F, Longley P A (2007) Geospatial analysis: A comprehensive guide to principles, techniques and software tools", 2nd edition, Troubador, UK available free online at: (1)

- Joel de la Cruz Villar “PHP y MySQL” Editorial Megabyte Lima-Perú 2006

- Wise, S. (2002) *GIS Basics*. London: Taylor & Francis.

- Wheatley, David and Gillings, Mark (2002) *Spatial Technology and Archaeology. The Archaeological Application of GIS*. London, New York, Taylor & Francis.

- Worboys, Michael, and Matt Duckham. (2004) *GIS: a computingperspective*. Boca Ratón: CRC Press.

- (Zavala 2000)Zavala R. 2000. *Diseño de un Sistema de Información Geográfica sobre internet*. Tesis de Maestría en Ciencias de la Computación. Universidad Autónoma Metropolitana-Azcapotzalco. México, D.F. En prensa.

- www3.uji.es/~mmarques/f47/apun/node68.html.
- www.programacion.com/bbdd/articulo/bbdd_disenyo/
- http://www.cindoc.csic.es/isis/03-1.htm
- http://faea.uncoma.edu.ar/materias/tdbd/
- http://www.itlp.edu.mx/publica/tutoriales/basedat2/
- Core JavaScript Reference – http://research.nihonsoft.org/javascript/CoreReferenceJS15/index.html
- XMLHttpRequest – http://developer.apple.com/internet/webcontent/xmlhttpreq.html
- HTML DOM – http://www.w3schools.com/htmldom
- Javascript – http://en.wikipedia.org/wiki/Javascript
- http://www.info-ab.uclm.es/asignaturas/42530/pdf/M1tema2.pdf
- http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software
- http://www.monografias.com/trabajos28/proyecto-uml/proyecto-uml.shtml
- (OMG 1999)Object Management Group. 1999. OMG Unified Modeling Language Specification (Draft). Versión 1.3. alfa R5, marzo de 1999. Disponible en: http://www.rational.com/uml
- (www.mkp.com/books_catalog/1-55860-465-0.asp).
- Publications (www.sigs.com/omo/articles/ambler.html)