Centro de Investigación y de Estudios Avanzados del IPN

Departamento de Ingeniería Eléctrica
Sección de Computación

Análisis Comparativo de Técnicas, Metodologías y Herramientas de Ingeniería de Requerimientos

Tesis que presenta

Juan Carlos Medina Martínez

Para obtener el grado de

Maestro en Ciencias en la especialidad de Ingeniería Eléctrica Opción Computación

Director de Tesis

Dr. Pedro Mejía Álvarez

México, D.F. Junio de 2004

Resumen

La aplicación de la Ingeniería de Software en el proceso de desarrollo de un sistema es un mecanismo que se emplea para garantizar la calidad de un producto de software. Una de las etapas importantes del proceso de desarrollo es la ingeniería de requerimientos, etapa en donde se definen inicialmente las características y restricciones con las que debe contar el sistema en desarrollo. En este trabajo de tesis se describen las actividades del proceso de Ingeniería de Requerimientos así como las tareas que comprenden cada actividad. Se presenta un análisis de técnicas, métodos y herramientas más recientes utilizadas en la ingeniería de requerimientos. Se ilustran las ventajas y desventajas que podemos encontrar en la utilización de estas técnicas y herramientas para cada actividad de este proceso, su aplicación en un caso de estudio y finalmente la definición del Documento de Especificación de Requerimientos.

Abstract

The application of Software Engineering in the process of development of a software system must be used to guarantee the quality of a software product. One of the most important stages of the development process is Requirements Engineering (RE), where the user need's, the initial specifications, characteristics and restrictions of the software system are defined. In this thesis, we intend to survey the most important activities of the Requirements Engineering process, as well as the tasks and tools that each activity includes. A comparative analysis of RE techniques, methods and tools is developed. Finally, in this thesis we developed a case study, were the requirements engineering processes is applied to model the application.

Agradecimientos

A CONACyT, por el apoyo económico otorgado mediante una beca que me permitió culminar satisfactoriamente mis estudios de maestría.

Al CINVESTAV-IPN, por darme la oportunidad y las facilidades para realizar la maestría en el Departamento de Ingeniería Eléctrica, Sección Computación.

A la UAG, por el apoyo otorgado para lograr mis estudios de maestría y superación académica.

Al Dr. Pedro Mejía Álvarez, por sus asesorías en la dirección de este trabajo de tesis, además de sus consejos y motivación en los momentos difíciles no sólo académicos sino de mi vida, gracias por ser un amigo más que un asesor.

Al Dr. Sergio V. Chapa Vergara y Adriano de Luca, por el apoyo y los comentarios brindados en la revisión de esta tesis.

A Sofía Reza, secretaria de la sección, quien siempre me brindo su apoyo en los tramites administrativos en el transcurso de mis estudios de maestría. Gracias por su amistad.

Al personal de la biblioteca de Ingeniería Eléctrica, Raúl Montaño Molina y Graciela Meza, quienes siempre mostraron su mayor disposición y apoyo cuando les solicitaba material bibliográfico.

A mis compañeros y amigos del CINVESTAV,

José Antonio,	Jaime,	Joselito,	Ulises,
Jorge,	Pedro,	José Trujillo,	Oscar,
Paola,	Laura,	Leticia,	Gregorio,
José Guadalupe,	Giner,	Sergio.	

Y a todas aquellas personas que me brindaron su amistad desinteresadamente e hicieron posible una estancia agradable en la Ciudad de México.

Dedicatorias

Le dedico este trabajo a una gran persona que siempre ha estado conmigo, Elvia, mi esposa, por su amor y comprensión, por su apoyo y creer que si podía lograr mi maestría, por sus palabras de aliento en los momentos difíciles. Gracias “chaparrita”.

A mi nena Katy, le pido disculpas por haberme perdido muchas de sus travesuras, espero que algún día comprenda el motivo por el cual pasó momentos sin su papá. Te quiero mucho mi nena.

A mis Padres, por su amor incondicional y la preocupación que siempre han mostrado cuando uno de sus hijos esta lejos de casa. Por impulsarme siempre a salir adelante, este logro también es de ustedes, los quiero mucho.

A mis hermanos, que siempre me han apoyado en los momentos difíciles y decisivos de mi vida.

A la familia Moreno Morales, que también es mi familia, gracias por su apoyo, sin ustedes no habría logrado esta meta.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

Introducción

Actualmente los sistemas computacionales están presentes en todas las áreas del conocimiento, por lo que estos sistemas deben garantizar la confiabilidad de los datos y de la información producida. De la información producida por los sistemas, los usuarios toman decisiones y acciones que afectan directamente los beneficios de su negocio. Es preciso señalar que el desarrollo de software debe ser un proceso controlado y planeado, en donde se toman en cuenta las necesidades y sugerencias de los usuarios y clientes que solicitan lo que el sistema debe y no debe hacer. Los errores en el software pueden causar daños o pérdidas en la información generada, y esto trasciende en las decisiones que tomen los usuarios del sistema. Las decisiones que se tomen con información errónea pueden producir bajas en el negocio, daños en el medio ambiente o hasta provocar la muerte de los usuarios en sistemas de alta criticidad. La ingeniería de software es una disciplina que proporciona a los desarrolladores y creadores de software, un conjunto de procedimientos y técnicas para llevar a cabo la especificación, análisis, diseño, implementación, validación e incluso el mantenimiento de software. Con la aplicación de la ingeniería de software se pueden reducir riesgos de fallos en un sistema en desarrollo e incrementar la posibilidad de la entrega del producto en el tiempo estimado, con calidad y dentro de los costos presupuestados. Generalmente las etapas utilizadas en el desarrollo de software son: Ingeniería de requerimientos, Diseño del sistema, Implementación, Validación y Mantenimiento.

Una etapa muy importante del proceso de ingeniería de software, es la ingeniería de requerimientos, la cual define el software que se desea producir y sus especificaciones. Este proceso se realiza mediante la obtención, el análisis, la especificación, la validación y la

administración de los requerimientos de software. Los requerimientos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema en desarrollo y las restricciones en las que debe operar. El resultado del proceso de requerimientos es la base para el diseño, la implementación y la evaluación del software. De esta forma, si no se descubren y especifican todos los requerimientos del sistema en desarrollo, podría conducirnos a la entrega de un producto de software incompleto, con alto índice de riesgos y poco funcional.

1.1 Motivación

La razón de la elección de este tema se fundamentó en:

- La escasa importancia que se le da a la Ingeniería de Software y en especial a la ingeniería de requerimientos en el proceso de producción de software.
- El gran número de proyectos de software que son abandonados en pleno proceso de desarrollo o son entregados con retrasos y con costos sobregirados.
- La existencia de un gran número de métodos y técnicas que lejos de ser un estándar confunden más a los desarrolladores y dificultan la definición de los requerimientos.
- La dificultad que existe para capturar requerimientos de software, con las técnicas existentes.
- La no existencia de documentos donde se propongan las técnicas que sirvan para desarrollar exitosamente las actividades de la ingeniería de requerimientos.

1.2 Planteamiento del problema

La ingeniería de requerimientos es una disciplina donde se establece un conjunto de actividades que son utilizadas para obtener, analizar, especificar, validar y administrar las necesidades del usuario, conocidas como requerimientos del sistema. Este proceso utiliza una combinación de técnicas, métodos y herramientas para obtener como resultado final el documento de especificación de los requerimientos. Dicho documento deberá contener una especificación completa, consistente y no ambigua del funcionamiento del sistema, utilizando lenguajes, diagramas y modelos de especificación de requerimientos. El documento de especificación de

requerimientos, sirve para negociar y validar los requerimientos de los usuarios, así también, como un contrato formal entre el cliente y el equipo de desarrollo. Este documento permite garantizar las características y funcionalidades del producto de software terminado.

En este trabajo de investigación, se identificarán y definirán las actividades de la ingeniería de requerimientos, las técnicas o métodos para la obtención, validación y administración de los mismos, se definirán los diferentes tipos de requerimientos que se encuentran en un proyecto de software. Se realizará la especificación de los requerimientos de un caso de estudio usando las técnicas y herramientas definidas para este proceso, y se concluirá con un análisis comparativo de las ventajas observadas sobre la aplicación de dichas técnicas y herramientas en cada una de las actividades de la ingeniería de requerimientos de software.

1.3 Objetivos de la Tesis

1.3.1 Objetivo General

Obtener un análisis comparativo, como resultado de la investigación, evaluación y aplicación de las diversas técnicas actuales sobre la ingeniería de requerimientos para el caso de estudio propuesto: el Sistema de Inscripción Virtual a cursos de la sección de computación de Ingeniería Eléctrica del CINVESTAV (SIV).

1.3.2 Objetivos Particulares

- Proponer una metodología para el desarrollo del proceso de Ingeniería de Requerimientos.
- Definir las actividades que se desarrollan en el proceso de la Ingeniería de Requerimientos.
- Demostrar el protagonismo que tiene el usuario/cliente en la definición de los requerimientos del sistema.
- Demostrar la importancia de la ingeniería de requerimientos dentro del proceso de producción de un sistema de software.
- Presentar y evaluar las diferentes maneras que existen para la identificación y especificación de los requerimientos.

- Identificar las ventajas y diferencias que existen entre las técnicas utilizadas en la ingeniería de requerimientos.
- Evaluar las herramientas existentes para la especificación y administración de requerimientos de software.

1.4 Organización de la Tesis

La tesis esta estructurada de la siguiente manera:

En el Capítulo 2 se definen conceptos de Ingeniería de Software que proporcionan un contexto general y permiten la ubicación del tema central de esta investigación. En el capítulo 3 se describe el proceso de Ingeniería de Requerimientos, se definen las actividades y tareas que se deben realizar para obtener la especificación de los requerimientos del sistema, se describen los diferentes tipos de requerimientos que se encuentran en el desarrollo del sistema y finalmente se define el esquema que debe tener un documento de especificación de requerimientos. En el capítulo 4 se realiza una revisión de las técnicas para obtención, análisis, especificación y validación de los requerimientos del sistema, así también un estudio de metodologías que se utilizan en el desarrollo de la Ingeniería de Requerimientos. En el capítulo 5 se presenta un estudio de las herramientas de software para la administración del Documento de Especificación de los Requerimientos. En el capítulo 6 se presenta un análisis de las técnica y metodologías definidas en el capítulo cuatro, además de una evaluación de las herramientas estudiadas en el capitulo cinco. En el capitulo 7 se presenta el caso de estudio (SIV) y la aplicación de las actividades de la Ingeniería de Requerimientos, así como el resultado obtenido, que es el documento de especificación de requerimientos de software para el SIV. En el capitulo 8 se presentan las conclusiones de esta investigación y los trabajos futuros a desarrollar.

Capítulo 2

Conceptos básicos de Ingeniería de Software

La construcción de software requiere de la aplicación de un proceso de desarrollo que permita garantizar la calidad de los productos generados y que satisfaga las necesidades de las personas que los usaran. Generalmente, se aplica un enfoque de la Ingeniería de Software para producir software y el proceso de desarrollo a utilizar depende del contexto del sistema que se este construyendo. Los productos que se obtienen a través de este proceso son documentos, programas y archivos de datos que producen los Ingenieros de Software mediante las actividades de la ingeniería de software. Podemos enfatizar que el proceso, las técnicas, los métodos y las herramientas comprenden la Ingeniería de Software. En este capítulo se establecen algunos conceptos importantes de Ingeniería de Software que nos permitan comprender y ubicar las actividades de la Ingeniería de Requerimientos en el proceso de desarrollo del software.

2.1 ¿Qué es la Ingeniería de Software?

Actualmente los sistemas computacionales están presentes en todas las áreas del conocimiento, por lo que estos sistemas deben garantizar la confiabilidad de los datos almacenados y la información producida. Los errores en el software pueden causar defectos en los datos y en la información generada, así como en las decisiones que tomen los usuarios que operan el sistema. En consecuencia, los defectos de software pueden producir pérdidas económicas en el negocio, daños en el medio ambiente o incluso en sistemas críticos, las fallas pueden provocar la muerte

de las personas que hacen uso de estos sistemas. Es preciso señalar que el desarrollo de software debe ser un proceso controlado y planeado, en el que deben tomarse en cuenta las necesidades y sugerencias de los usuarios o clientes quienes finalmente son los que más conocen el funcionamiento de su negocio.

La utilización de modelos de representación en el proceso de desarrollo de software permite reducir la probabilidad de riesgos y fallas, además garantizan la entrega en tiempo y forma del sistema al cliente. Los productos son entregados con calidad y dentro de los costos estimados. Se puede definir a la Ingeniería de Software como una disciplina que proporciona un conjunto de procedimientos y técnicas para la producción y desarrollo sistematizado, disciplinado y cuantificable de aplicaciones de software, que satisface los requerimientos de funcionalidad y desempeño [1]. Generalmente como expertos en el área de la computación nos dedicamos a analizar problemas relacionados con un sistema de computación existente, por lo tanto, es necesario estudiar el ambiente del sistema, para no imponer técnicas de computación en todo problema que surja. Primero debe resolverse el problema, después si es necesario, se puede utilizar la tecnología computacional como herramienta para implementar nuestra solución.

2.2 ¿Qué hacen los Ingenieros de Software?

Las computadoras y los lenguajes de programación, pueden verse desde dos puntos de vista: como objetos de estudio y como herramientas a utilizar en el diseño e implementación de una solución a un problema. Los Ingenieros de Software, en lugar de investigar el diseño del hardware ó de probar teorías acerca de cómo trabajan los algoritmos, trabajan con las funciones de un sistema de cómputo como parte de una solución general. Utilizan todos sus conocimientos técnicos de computación para tratar de resolver problemas previamente estudiados y que pueden ser factiblemente computarizados mediante técnicas y herramientas de la Ingeniería de Software. El desarrollo de un sistema involucra a personas que son integradas en equipos y que desarrollan tareas específicas. Se conoce con el nombre de **desarrolladores** a los ingenieros de software debido a que desempeñan diferentes papeles en el proceso de desarrollo de software.

En el equipo de desarrolladores de software se pueden encontrar ingenieros especializados en cada una de las actividades, por lo tanto deben existir **analistas de requerimientos**, quienes trabajan con los clientes desglosando en requerimientos las necesidades de los clientes. Una vez conocidos y documentados los requerimientos, los analistas de requerimientos trabajan con los

diseñadores para generar una descripción a nivel de sistema de lo que el sistema debe hacer. Los diseñadores trabajaran con los **programadores** para describirles el sistema en una forma tal que dichos programadores implementen lo que los requerimientos especifican. Posteriormente, el código debe ser probado, y en ocasiones las primeras pruebas la hacen los mismos programadores, aunque a veces intervienen **probadores** adicionales para ayudar a detectar defectos que los programadores pasan por alto. Después de comprobarse la funcionalidad y calidad del sistema se trabaja con el cliente para verificar que el sistema completo resulta en lo que el cliente desea. Esto se hace comparando cómo trabaja el sistema en relación al conjunto inicial de requerimientos. Finalmente, los **entrenadores** enseñan al usuario como se utiliza el sistema.

Para muchos sistemas de software, la aceptación por parte del cliente no significa la finalización de las tareas de desarrollo, si existen defectos después de la aceptación del sistema, intervendrá un **equipo de mantenimiento** para corregirlos. De cualquier forma, los requerimientos del cliente pueden cambiar a medida que transcurre el tiempo y deberán hacerse los cambios correspondientes al sistema. El mantenimiento puede implicar a analistas para determinar qué requerimientos se agregan o cambian, a diseñadores para establecer en qué lugares del diseño del sistema deben hacerse los cambios, a programadores para implementar los cambios, a probadores para asegurar que el sistema cambiado funciona correctamente y a entrenadores para explicar a los usuarios la forma en que los cambios afectan el uso del sistema [1].

2.3 El Software

Cuando hablamos de software, muchas veces nos referimos a programas de computadoras, pero el software no sólo son programas, sino todos los documentos generados, la configuración ó la estructura de los datos que se necesitan para hacer que estos programas funcionen de manera correcta. En general, un sistema de software consiste de varios programas independientes, archivos de configuración, documentación que describe la estructura del sistema, planes y manuales de usuario que explica como usar el sistema. En la literatura de Ingeniería de Software [2], se distinguen dos tipos de productos de software.

1. **Software genérico**. Son sistemas aislados producidos por una organización de desarrollo que se vende al mercado abierto a cualquier cliente que le sea posible adquirirlo. La organización de desarrollo controla la especificación del producto.

2. **Software personalizado**. Son sistemas requeridos por un cliente en particular. Una empresa de software desarrolla un sistema especial para un cliente. Por lo general, la especificación de este producto es controlada por el cliente del software, y los desarrolladores deben trabajar con esa especificación.

Las aplicaciones de software ya sean genéricas o especializadas, se encuentran dentro de alguna de las siguientes clasificaciones [4].

a. **Independientes**. Es software que reside en una sola computadora, además de que no se conecta con otro software o hardware.

b. **Inmersos**. El software es parte de una aplicación única que incluye hardware.

c. **De tiempo real**. La aplicación de software debe ejecutar funciones en un límite de un tiempo corto, generalmente en milisegundos.

d. **De redes**. El software consiste en componentes que interactúan utilizando los beneficios de una red.

2.4 Atributos de un buen software

Un software de alta calidad es aquel que satisface las necesidades del usuario y posee las siguientes características:

- **Útil y aprovechable**. Es decir, hace el trabajo de los usuarios más fácil o mejor.
- **Fiable**. Un software debe contener pocos errores o idealmente ninguno.
- **Flexible**. En el transcurso del desarrollo del software, las necesidades y requerimientos de los usuarios cambian, estos cambios deben poder realizarse en el momento que se requieran. A los cambios que son realizados después de la entrega del sistema, se les conoce como mantenimiento.
- **Accesible**. La venta como el costo del mantenimiento debe estar en función de la mano de obra y el alcance del sistema.
- **Disponible**. La disponibilidad del software consiste en dos aspectos: primero, tiene que poder ejecutarse con el equipo de cómputo disponible y debe ser suficientemente portable. Segundo, un proyecto de software debe culminar con la entrega del sistema prometido.

2.5 Defecto, error y falla en el software

A pesar del gran avance que hay en las técnicas para la Ingeniería de Software siguen existiendo sistemas que presentan problemas. Los defectos en el software en algunos casos solo causan molestias, pero en otros cuestan grandes cantidades de tiempo y dinero, y en ocasiones son una amenaza para la vida. En la figura 2.1 se muestra la relación que existe entre defectos, errores y fallas.

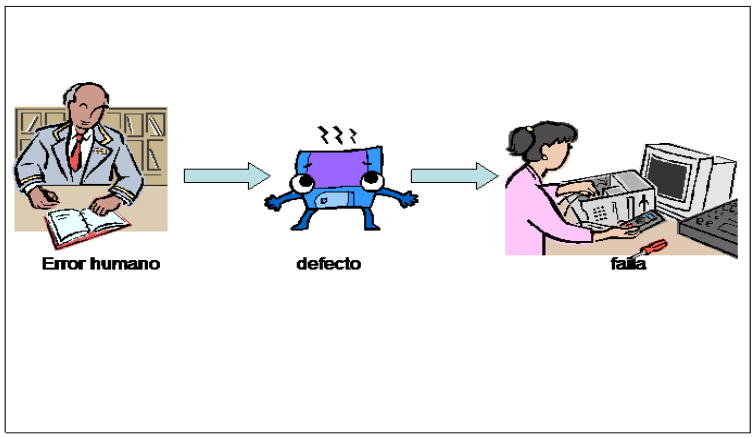


Figura 2.1 Relación entre defecto, error y falla.

Se produce un **defecto** cuando una persona tiene una equivocación denominada **error**, al realizar alguna actividad concerniente al software. Por ejemplo, un diseñador puede comprender mal un requerimiento y crear un diseño que no corresponda con la interpretación del analista y de las necesidades del usuario, de esta forma, el defecto en el diseño puede propagarse en el momento de que los programadores traduzcan el diseño a código. Esto puede dar lugar a otros defectos, tales como código incorrecto y una descripción incorrecta en el manual del usuario. De esta manera un error puede generar múltiples defectos y un defecto puede encontrarse en cualquier producto en desarrollo o mantenimiento.

Una **falla** es un desvío respecto al comportamiento requerido del sistema, puede descubrirse antes o después de la entrega del sistema, durante la prueba, la operación o durante el mantenimiento. Una falla indica que el sistema no esta funcionando como se ha especificado [1].

Un defecto es una vista interna del sistema desde la perspectiva de los desarrolladores, mientras que una falla es una vista externa o un problema que ve un usuario, sin embargo, no todos los defectos llevan a producir una falla. Por lo tanto, deben buscarse mecanismos que minimicen el número de errores en la etapa de especificación de los requerimientos para evitar futuros defectos y fallas. Esto se puede lograr empleando técnicas, métodos y herramientas adecuadas al dominio del sistema en el proceso de desarrollo del software.

2.6 Elementos en el desarrollo de software

La elaboración de un **proyecto** de software, implica la intervención de diferentes tipos de **personas** guiadas mediante un **proceso** de desarrollo para obtener un **producto** de software. Generalmente, en el proceso de desarrollo se utiliza un conjunto de **herramientas** para automatizar el proceso y facilitar las tareas al equipo de desarrollo. Podemos decir que estos son los elementos principales que intervienen en la manufactura de software. [5]

1. **Personas**. Son el elemento principal y son los autores de un proyecto de software y generalmente comprende a los arquitectos, analistas, desarrolladores, Ingenieros de prueba y al personal de soporte, además de los clientes, usuarios y otros interesados.
2. **Proyecto**. Es el elemento organizativo mediante el cuál se gestiona el desarrollo de software. El resultado es la versión final de un producto.
3. **Producto**. Comprende a todos los artefactos que se crean durante el desarrollo del proyecto (documentos, modelos, diagramas, código fuente, código ejecutable y pruebas).
4. **Proceso**. Son un conjunto de actividades necesarias para transformar las necesidades del usuario en un producto.
5. **Herramientas**. Es el software que se utiliza para automatizar las actividades del proceso.

2.7 El proceso de desarrollo de Ingeniería de Software

El **proceso de desarrollo de software** es una descripción de la construcción del software que contiene actividades organizadas de modo que en conjunto producen código probado. No existe una definición estándar de estas actividades y muchos autores le dan importancia a algunas más que a otras. Generalmente, podemos clasificar estas actividades en:

1. Ingeniería de requerimientos
2. Diseño del sistema
3. Implementación del software
4. Prueba o validación del software
5. Mantenimiento

En la **Ingeniería de requerimientos**, es necesario conocer la naturaleza del sistema, entender lo que desean los clientes, delimitar el alcance del sistema teniendo en cuenta el tiempo disponible, el presupuesto y el personal asignado. El producto generado de esta etapa, es un **documento de especificación de requerimientos** en donde se encuentran definidos todos los servicios requeridos del sistema y las restricciones sobre las que debe operar.

El **diseño del sistema** toma como base el documento de especificación de requerimientos, agrega detalles a cada requerimiento, identifica los subsistemas, describe la estructura interna de los datos y las interfaces entre los componentes del sistema. El diseño del sistema utiliza varios modelos para la representación del sistema desde diferentes perspectivas y niveles de abstracción. El resultado de esta etapa es la especificación precisa de los algoritmos y estructuras de datos que van a implementarse. En la etapa de **implementación** se lleva a cabo la codificación del sistema, se deben satisfacer los requerimientos de la manera que especifica el diseño detallado. El programador examina los documentos generados en las etapas anteriores para evitar inconsistencias entre los documentos. Se toman en cuenta los estándares de programación, así como los lenguajes de programación. Las **pruebas y la validación** se utilizan para demostrar que el sistema cumple con su especificación y satisface las necesidades del usuario. De cualquier forma, es necesario llevar a cabo un proceso de verificación por medio de inspecciones y revisiones desde la especificación de los requerimientos hasta la puesta en marcha del sistema.

El propósito de realizar pruebas no es demostrar que una aplicación es satisfactoria, sino determinar con firmeza en qué parte no lo es. Las pruebas nos permiten mostrar la presencia de defectos en el sistema. El **mantenimiento** de software consiste en las actividades realizadas sobre el sistema una vez entregado y puesto en marcha. Una definición es [6]. "El proceso de modificar un sistema o componente de software entregado para corregir defectos, mejorar el desempeño, mejorar algún atributo, o adaptarlo al cambio de entorno". Como se observa, el resultado de la Ingeniería de Software consiste en mucho más que el código del sistema, es decir, incluye planes, informes, documentos e inclusive programas llamados prototipos que son desechados después de lograr el objetivo por lo que fueron creados.

2.8 Paradigmas del proceso de desarrollo de software

Un paradigma o modelo del proceso de desarrollo de software, es una descripción de las actividades que se llevan a cabo en la elaboración de un producto de software. Al proceso completo de desarrollo, también se le conoce como **ciclo de vida del software**, debido a que en él se detalla la vida de un sistema desde su concepción, implementación, entrega, utilización y hasta su mantenimiento. En la literatura de Ingeniería de Software existe una variedad de modelos o paradigmas de desarrollo. Los modelos más utilizados se describen a continuación.

2.8.1 Modelo de desarrollo en cascada

El "modelo en cascada", define un enfoque secuencial del proceso de desarrollo, separando en etapas y cada etapa es fundamental en el desarrollo (figura 2.2). Cada etapa debe completarse antes de comenzar la siguiente etapa. El problema que presenta este modelo es debido a la separación de las actividades en etapas, en cada fase el resultado es uno o más documentos aprobados. En la práctica estas actividades del desarrollo de un producto de software están entrelazadas y requieren de continuas iteraciones, además si un error no es detectado al principio de la etapa, puede ser desastroso encontrarlo en etapas posteriores. Es difícil dejar definidos los requerimientos en la primera etapa del desarrollo, como lo expresa el modelo, esto podría generar problemas a la hora de encontrar nuevos requerimientos en las últimas etapas y no fuera posible regresar a la primera etapa para agregarlos. Para el modelo en cascada el cliente podría tener una versión operativa del sistema sólo hasta alcanzar las etapas finales del desarrollo [2].

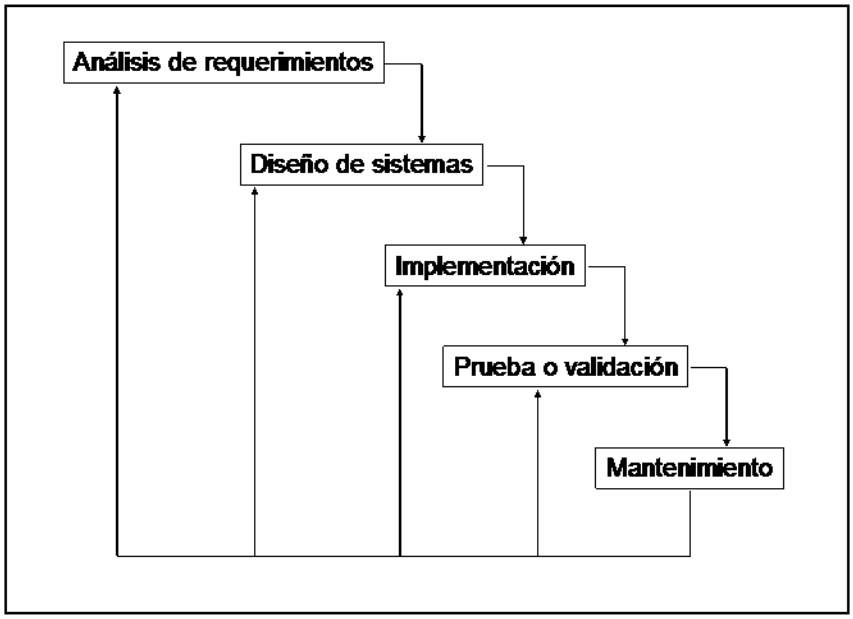


Figura 2.2 Modelo en cascada.

2.8.2 Desarrollo Evolutivo

Este paradigma recomienda el desarrollo de una implementación inicial del sistema, que debe ser presentada al cliente para su negociación, y posteriormente refinada mediante versiones hasta alcanzar el desarrollo completo del sistema. Este modelo de desarrollo es más conocido como **prototipado**, en donde las actividades son llevadas a cabo en forma concurrente y tienen retroalimentación en todo el proceso.

Existen dos tipos de desarrollos evolutivos:

1. **Desarrollo Exploratorio**. En este tipo de desarrollo se trabaja con el cliente para encontrar sus requerimientos y entregar un sistema final. El sistema es mejorado cada vez que existan nuevas propuestas del cliente.
2. **Prototipos Desechables**. El objetivo de los prototipos desechables es comprender los requerimientos del cliente. Una vez logrado el objetivo se desecha el prototipo actual para construir otro que permita comprender mejor los requerimientos.

La principal ventaja de este enfoque se basa en que la especificación puede desarrollarse en forma creciente, conforme los usuarios adquieran un mayor conocimiento de su problema. Sin embargo, este enfoque también presenta algunas deficiencias:

- **El proceso no es visible**. Se tienen que realizar entregas regulares de versiones del sistema para medir el progreso, y es muy costoso producir documentos para cada versión del sistema.
- Generalmente se tiene una **estructura deficiente**. Los cambios continuos en el software tienden a descomponer la estructura del sistema, además su aplicación es una tarea difícil y costosa.
- Se requieren **herramientas y técnicas especiales** que permitan un desarrollo rápido y que no presenten incompatibilidades entre sí.

2.8.3 Modelo de desarrollo Incremental e Iterativo

Una manera de reducir el tiempo de entrega de un sistema, es mediante la utilización del desarrollo por fases. El sistema es diseñado de tal forma que pueda ser entregado por módulos, lo que permite que los usuarios dispongan de su parcial funcionalidad a medida que el sistema se encuentra en desarrollo [1]. Los dos enfoques más conocidos de este modelo son:

1. **Desarrollo incremental**
2. **Desarrollo iterativo**

En el **Desarrollo incremental** el sistema es fragmentado en subsistemas de acuerdo a su funcionalidad y de como este definido en el documento de requerimientos. Las versiones son definidas iniciando con un módulo funcional pequeño y con cada nueva versión se agrega funcionalidad al sistema total (figura 2.3). Uno de los problemas que presenta este enfoque es la dificultad en la identificación de los recursos comunes que requieren todos los incrementos. Esto se debe a que los requerimientos no se definen en detalles hasta que un incremento se implemente.

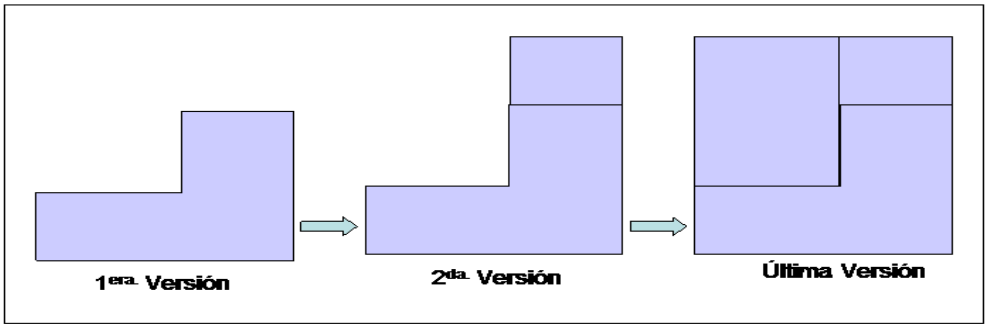


Figura 2.3 Modelo incremental.

En el **Desarrollo iterativo** el sistema es entregado completo, aunque la funcionalidad sea primitiva, es decir, que las partes del sistema no este funcionando totalmente. Posteriormente cada parte del sistema es reforzado con calidad e implementándose las funcionalidades faltantes en las versiones anteriores (figura 2.4).

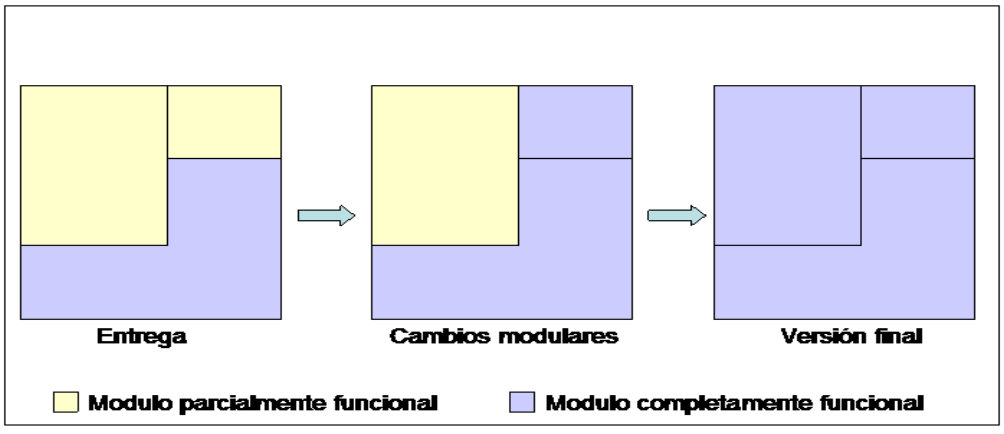


Figura 2.4 Modelo iterativo.

2.8.4 Desarrollo en espiral

El modelo en espiral fue propuesto por Boehm en el año de 1988. Este modelo representa el proceso de software como una espiral, examina el progreso de desarrollo de software tomando en consideración los riesgos involucrados para minimizarlos y controlarlos, de esta forma combina las actividades del desarrollo con la gestión del riesgo [1]. El espiral se divide en cuatro sectores que comprenden: la definición de objetivos, la evaluación y reducción de riesgos, el desarrollo y validación, y por último la planeación (figura 2.5).

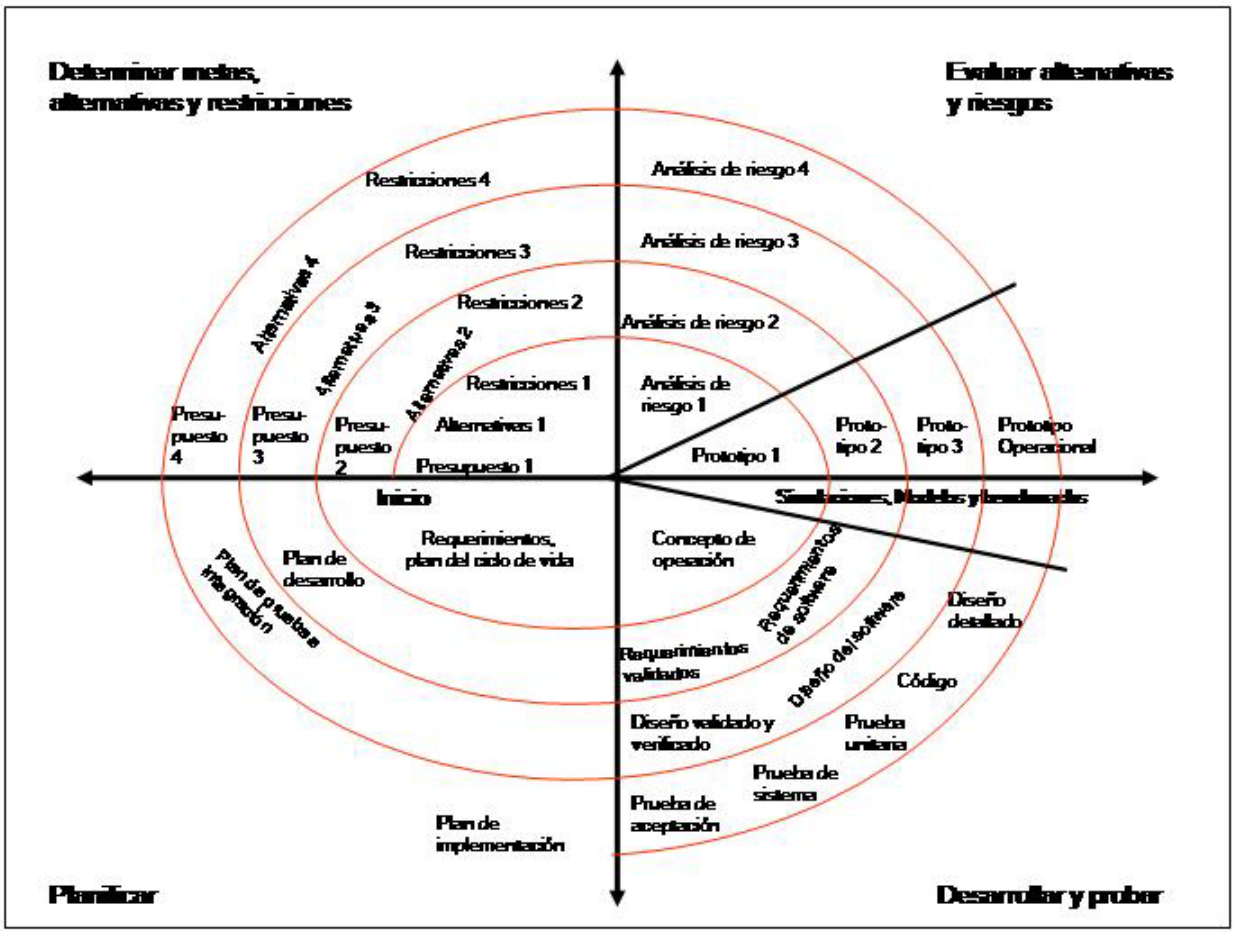


Figura 2.5 Desarrollo en espiral.

El modelo del espiral se ajusta al avance de los proyectos, sin embargo requiere de una administración mucho más cuidadosa que la del modelo en cascada. El ciclo del espiral comienza con los requerimientos y un plan de desarrollo, agregando una evaluación de riesgos y construye prototipos alternativos antes de escribir el documento de conceptos de operación. El documento de conceptos de operación describe en un alto nivel como debe trabajar el sistema, es decir, especifica un conjunto de requerimientos completos y consistentes. El principal producto de la primera iteración es el concepto de operaciones, en la segunda iteración son los requerimientos, en la tercera iteración el diseño y de la cuarta son las pruebas. Para cada iteración el análisis de riesgos pondera diferentes alternativas en base de los requerimientos y restricciones, el prototipado verifica el grado de factibilidad antes de elegir alguna alternativa de desarrollo en particular, si se identifican riesgos los lideres del equipo de desarrollo son los que deciden como eliminarlos o minimizarlos [1].

2.8.5 Modelos de Métodos Formales

Comprenden un conjunto de actividades que conducen a la especificación matemática del software. Los métodos formales permiten a los ingenieros de software especificar, desarrollar y verificar un sistema, aplicando una notación matemática.

El enfoque más conocido del proceso de desarrollo formal es “la ingeniería de la sala limpia” (clear room), desarrollado por IBM. A pesar de que estos métodos proporcionan mecanismos para reducir muchos de los problemas que son difíciles de superar con otros enfoques de la Ingeniería de Software, existen situaciones en torno a su aplicabilidad [3].

- El desarrollo de modelos formales actualmente es caro y consume mucho tiempo.
- La mayoría de los desarrolladores no tienen antecedentes necesarios para aplicarlos.
- No son recomendables utilizarlos como mecanismos de comunicación con los clientes del sistema, ya que no tienen conocimientos técnicos.

2.8.6 Desarrollo basado en componentes

Este enfoque se basa en la reutilización de componentes de software reutilizable. Generalmente esto sucede cuando los desarrolladores conocen diseños o código similares requeridos y que ya fueron elaborados para otros sistema [2]. Para garantizar el proceso de desarrollo, y además de las actividades de especificación de requerimientos y la validación del sistema, se integran nuevas actividades que son:

1. **Análisis de componentes**. En esta fase se buscan los componentes que satisfagan las especificaciones de los requerimientos del sistema a desarrollar.
2. **Modificación de componentes**. Se analizan tanto los requerimientos como los componentes, adaptando los componentes que lo requieran, y si no son posible estos cambios, se realiza nuevamente la búsqueda de nuevos componentes para buscar soluciones alternativas.
3. **Diseño de sistemas con reutilización**. En esta actividad, se diseña el sistema con los componentes encontrados y si no hay componentes disponibles se diseñan nuevos.
4. **Desarrollo e integración**. La integración de sistemas es parte del desarrollo y los componentes que no se puedan adquirir se desarrollan.

Con la reutilización podemos obtener ventajas en la reducción de la cantidad de software a desarrollar, de costos y riesgos, conduciendo a la entrega rápida del software. Sin embargo, al igual que todos los modelos, este también presenta inconvenientes. El principal inconveniente consiste en que la reutilización de componentes nos conduce a que un sistema no cumpla con las necesidades reales de los usuarios definidas en el documento de requerimientos, esto se debe a los ajustes que se les hace a los componentes para adaptarlos al sistema en desarrollo.

2.8.7 El Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Desarrollo de Software (por sus siglas en ingles RUP, Racional Unified Process), es un modelo propuesto por Booch, Jacobson y Rumbaugh, esta basado en componentes y utiliza como modelado el Lenguaje Unificado de Modelado (UML). Se basa sobre tres ideas básicas que son: casos de uso, arquitectura y desarrollo iterativo e incremental.

El proceso unificado se repite a lo largo de una serie de ciclos que conforman la vida de un sistema (figura 2.6). Cada ciclo concluye con una versión del producto para los clientes. [5]

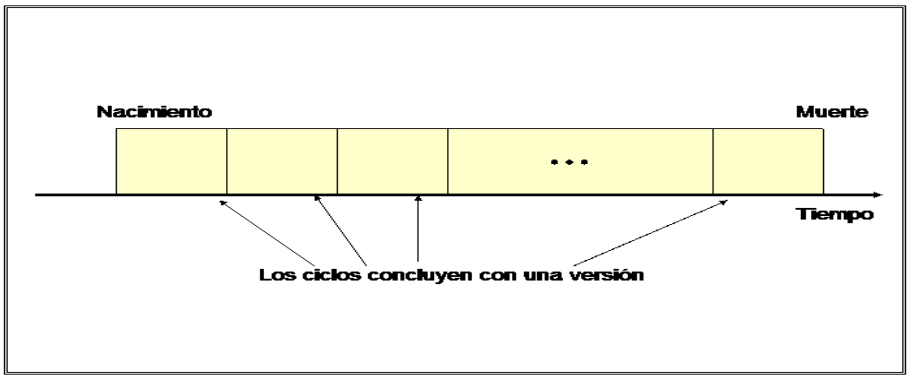


Figura 2.6 Ciclos de Proceso Unificado.

Cada ciclo consta de cuatro fases que son: inicio, elaboración, construcción y transición [5]. Cada fase consta de varias iteraciones, como se ilustra en la figura 2.7

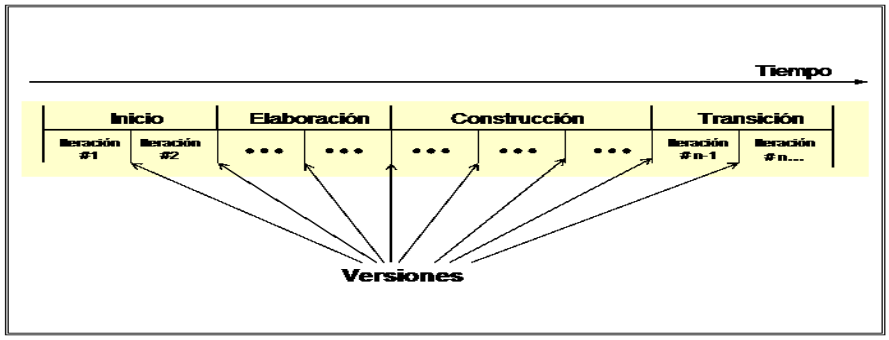


Figura 2.7 Fases en un ciclo del Proceso Unificado.

1. En la **fase de inicio** se define el problema o una necesidad y se evalúa su factibilidad y riesgos. Se definen un modelo de casos de uso simplificado que contenga los casos de uso más críticos.
2. Durante la **fase de Elaboración** se especifican con más detalles la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La arquitectura se expresa en forma de vistas de todos los modelos del sistema, los cuales juntos representan el sistema

entero. Además, en esta fase se planifican las actividades y se estiman los recursos necesarios para concluir el proyecto.

3. En la **fase de construcción** se crea el producto, la arquitectura crece hasta convertirse en el sistema completo, el producto crece en su forma básica, pero todavía requiere trabajo para liberarlo.
4. La **fase de transición** corresponde a los procesos de instalación y mantenimiento, el producto es probado por un número reducido de usuarios con experiencia para buscar defectos y deficiencias. Los desarrolladores corrigen los problemas e integran algunas de las mejoras sugeridas en una versión final.

El proceso asume que los flujos de trabajos de **requerimientos**, **análisis**, **diseño**, **implementación** y **prueba** participan en cada una de las iteraciones. Sin embargo, estas actividades tienen diferentes necesidades específicas de cada fase como se muestra en la figura 2.8. El proceso unificado produce seis modelos o vistas de la aplicación [5] y son los siguientes:

1. Los requerimientos se capturan en el **modelo de casos** de uso.
2. El **modelo de análisis** describe el sistema como un conjunto de clases.
3. El **modelo de diseño** define la estructura del sistema como un conjunto de subsistemas e interfaces.
4. El **modelo de despliegue** define la distribución.
5. El **modelo de implementación** establece la correspondencia entre clases y los componentes.
6. El **modelo de pruebas** verifica que el sistema ejecutable proporcione la funcionalidad descrita en el modelo de casos de uso.

Todos los modelos están relacionados entre sí mediante dependencias de rastreabilidad hacia atrás y hacia delante.

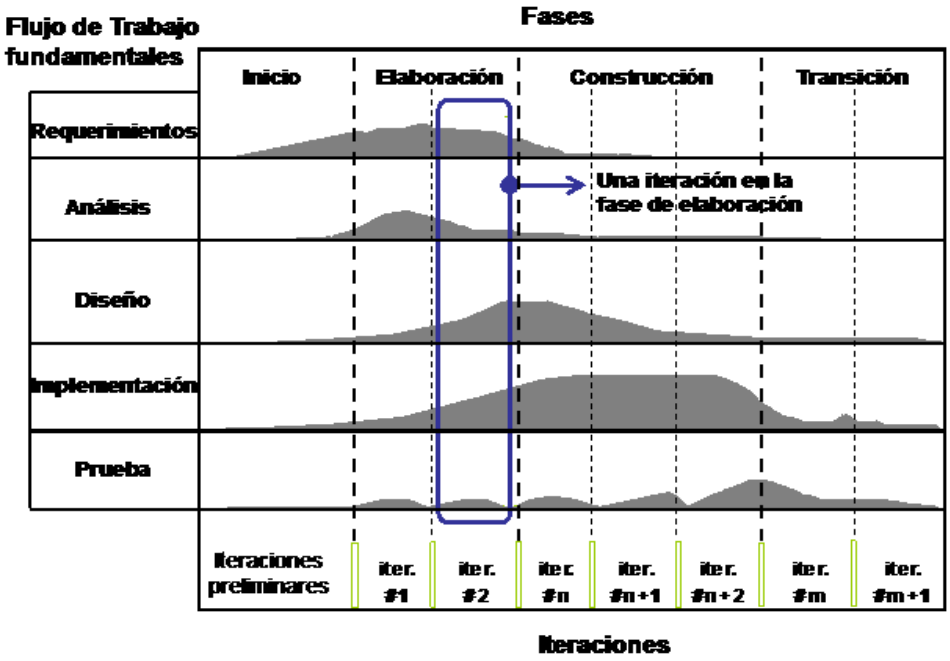


Figura 2.8 El Proceso Unificado de Desarrollo de Software.

2.9 Problemas que enfrenta la Ingeniería de Software

A pesar de los grandes avances que existen en la Ingeniería de Software, aún hay problemas que se presentan en el desarrollo de proyectos de software y que impiden su conclusión o retrasan su entrega. Algunos de los problemas que enfrenta la Ingeniería de Software son:

1. La necesidad de las empresas clientes, por presentar sus productos en el mercado en el menor tiempo posible.
2. La complejidad de los sistemas.
3. La dificultad de integrar cambios a sistemas con código obsoleto.
4. La difícil integración de un nuevo software a otros sistemas existentes.
5. La elaboración de software sin utilizar los procesos de desarrollo que garanticen la conclusión de los productos con calidad y sin alterar los costos con que fueron concebidos.
6. La introducción de nuevas técnicas y herramientas en el proceso de desarrollo de un sistema y que requieran de amplios conocimientos técnicos para su manejo.
7. El no descubrimiento de todos los requerimientos de software en etapas iniciales del desarrollo del sistema.
8. La naturaleza de cambio en que se encuentran las empresas clientes y que requiere que los sistemas se adapte a estos cambios.
9. La mala interpretación de los requerimientos por los desarrolladores.
10. La falta de entendimiento entre el equipo de desarrollo y los usuarios o clientes para homogenizar la comprensión de los requerimientos.

Estos son algunos de los problemas que podemos encontrar y que pueden ser los responsables del fracaso de muchos proyectos de software.

2.10 Errores históricos del Software

En la literatura de Ingeniería de Software se encuentran registradas una gran cantidad de citas que hablan de proyectos de software que no fueron culminados, software con entregas retrazadas y con presupuestos mayores a los estimados e incluso de fallos drásticos en los sistemas que si fueron entregados. Entre estas citas encontramos los siguientes.

- **El vuelo del Ariane 5**. Se trata de un vuelo inaugural realizado el 4 de julio de 1996 que terminó en una explosión en la pista debido a un error de software en el sistema de referencia inercial. Esto fue causado por la conversión de un número de punto flotante de 64 bits a un entero con signo de 16 bits, relacionado a la velocidad horizontal del cohete con respecto a la plataforma y que había sido utilizado en el modelo anterior del Ariane 5 [6, 7, 24, 25 y 26].
- **Sistema de Ambulancias en Londres**. Un software que fallo dos veces en el año de 1992, debido a fallas de Ingeniería de Software, específicamente defectos en el despachador automatizado, decisiones de administración, errores en el diseño y en la implementación del software. La falla provocó que las ambulancias no llegarán a tiempo donde se les requerían, se piensa que hubo gente que murió debido a esta falla en el software [6, 7, 27, 28, 29, 30 y 31].
- **La máquina Therac-25**. Se trata del mal funcionamiento en el software de la máquina de terapia de radiaciones Therac-25, donde seis personas sufrieron sobredosis de radiaciones, se cree que la mitad de estas personas murieron. La principal causa fue el desarrollo de un sistema complejo, la falta de calidad, no fue probado adecuadamente, la reutilización de código en un nuevo contexto y con poca documentación que generó que no se aplicará una adecuada corrección al fallo provocado [6, 7,32].
- **El sistema Taurus**. Un sistema de ajuste de transacciones automatizado planificado para la Bolsa de Londres. El sistema operó alrededor de cinco años, con un costo cerca de los 75 millones de libras y se estima que las pérdidas de los clientes fue aproximadamente de 450 millones de libras. Además, del daño incalculable a la reputación de la Bolsa de Londres [7].

- **Sistema de mantenimiento de equipaje de Denver**. Un gran sistema que implicaba aproximadamente 300 computadoras, se sobrecargó impidiendo abrir el aeropuerto a tiempo. Resulto contener demasiados errores y costo aproximadamente el 1.5% del valor inicial de cerca de 200 millones de dólares para corregirlo [7].

Estos son algunos de los casos más mencionados sobre fallos de sistemas, y que son generados por errores en algunas de las etapas del proceso de desarrollo de software.

Capítulo 3

Ingeniería de Requerimientos

Una de las primeras etapas del proceso de desarrollo de Software es la Ingeniería de Requerimientos, en esta fase se definen y especifican los requerimientos de los clientes y usuarios. Las actividades involucradas en la Ingeniería de Requerimientos son: la **obtención**, el **análisis**, la **especificación**, la **validación** y la **administración** de los requerimientos de software.

Los requerimientos de software se definen como las necesidades de los clientes, los servicios que el usuario desea que proporcione el sistema y las restricciones sobre las que el software debe operar. El resultado del proceso de requerimientos es la base para el diseño, la implementación y la evaluación del software. De esta forma, si no se descubren y especifican todos los requerimientos del sistema en desarrollo, podría conducirnos a la entrega de un producto de software incompleto, con alto índice de riesgos y poco funcional.

A pesar de los avances en la Ingeniería de Software, existe una gran cantidad de desarrolladores que no hacen uso de procedimientos y actividades que proporciona la Ingeniería de Software en el desarrollo de sistemas; esto provoca que un gran número de proyectos de software sean abandonados en pleno proceso de desarrollo o que sean entregados con retrasos y a costos mayores a los estimados. A esta problemática se le une el hecho de que existen una gran cantidad de métodos y técnicas, que lejos de ser estándares, confunden más a los desarrolladores y dificultan la definición de los requerimientos.

Las tareas más complejas para los ingenieros o especialistas de software en el proceso de Ingeniería de Requerimientos y son:

1. Tratar con la naturaleza del sistema y comprender su ambiente.
2. Encontrar los componentes y su interacción dentro del sistema.
3. Definir los servicios que el sistema debe ofrecer al usuario.
4. Definir las restricciones o limitantes del sistema.

Estas tareas definen los problemas de la Ingeniería de Requerimientos y que deben enfrentar los desarrolladores y analistas de software. El proceso de descubrir, analizar, documentar y verificar los servicios y restricciones del sistema, sólo se logra utilizando las técnicas definidas por la Ingeniería de Requerimientos.

3.1 El proceso de Ingeniería de Requerimientos

El proceso de Ingeniería de Requerimientos, involucra a todas las actividades necesarias para crear y mantener el documento de requerimientos del sistema. Para realizar este proceso, es necesario previamente obtener un análisis de factibilidad que indique si es factible continuar con el desarrollo del sistema, además de conocer las necesidades del cliente para tener un contexto general del sistema.

Las actividades que se definen en el proceso de la Ingeniería de Requerimientos son las siguientes:

1. Obtención de los requerimientos
2. Análisis de requerimientos
3. Especificación de requerimientos
4. Validación de documento de especificación
5. Administración de requerimientos

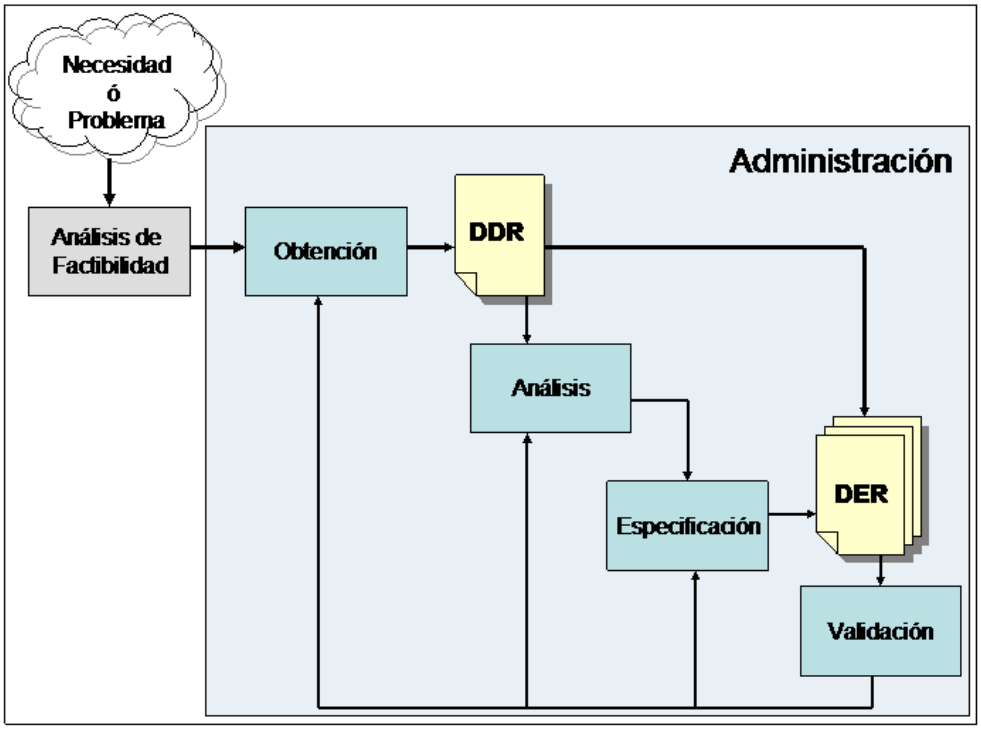


Figura 3.1 El proceso de Ingeniería de Requerimientos en el modelo de cascada.

En la figura 3.1 se muestra la interacción que existe entre las diferentes actividades del proceso de Ingeniería de Requerimientos utilizando el paradigma de cascada, en donde cada actividad tiene un regreso a la actividad anterior. Se parte del hecho de que existe una definición general del problema y de un análisis de factibilidad que indica la continuación del desarrollo del software. De esta manera, el proceso de Ingeniería de Requerimientos inicia con la actividad de la obtención de los requerimientos, mediante la revisión de la información existente en manuales y documentos y entrevistas a los usuarios para descubrir y analizar los requerimientos. La especificación y validación son actividades que pueden ser repetidas cuantas veces sean necesarias hasta lograr el nivel de refinamiento en que se deseen especificar los requerimientos. Finalmente cada cambio en los requerimientos, es administrado en versiones del Documento de Especificación de Requerimientos.

Otra manera de desarrollar el proceso de Ingeniería de Requerimientos es mediante el paradigma del espiral, donde las actividades son repetidas hasta cuando se obtenga un documento de requerimientos aceptable, como se muestra en la figura 3.2. Mientras existan problemas en una

versión previa del documento de requerimientos (borrador), se reingresa a las etapas del espiral, el proceso finaliza cuando se produzca un documento aceptable, de la misma forma, el proceso puede terminar por factores externos como la presión del cliente ó la falta de recursos.

Después de que el documento sea producido satisfactoriamente, cualquier cambio futuro en los requerimientos es parte de la administración de los requerimientos.

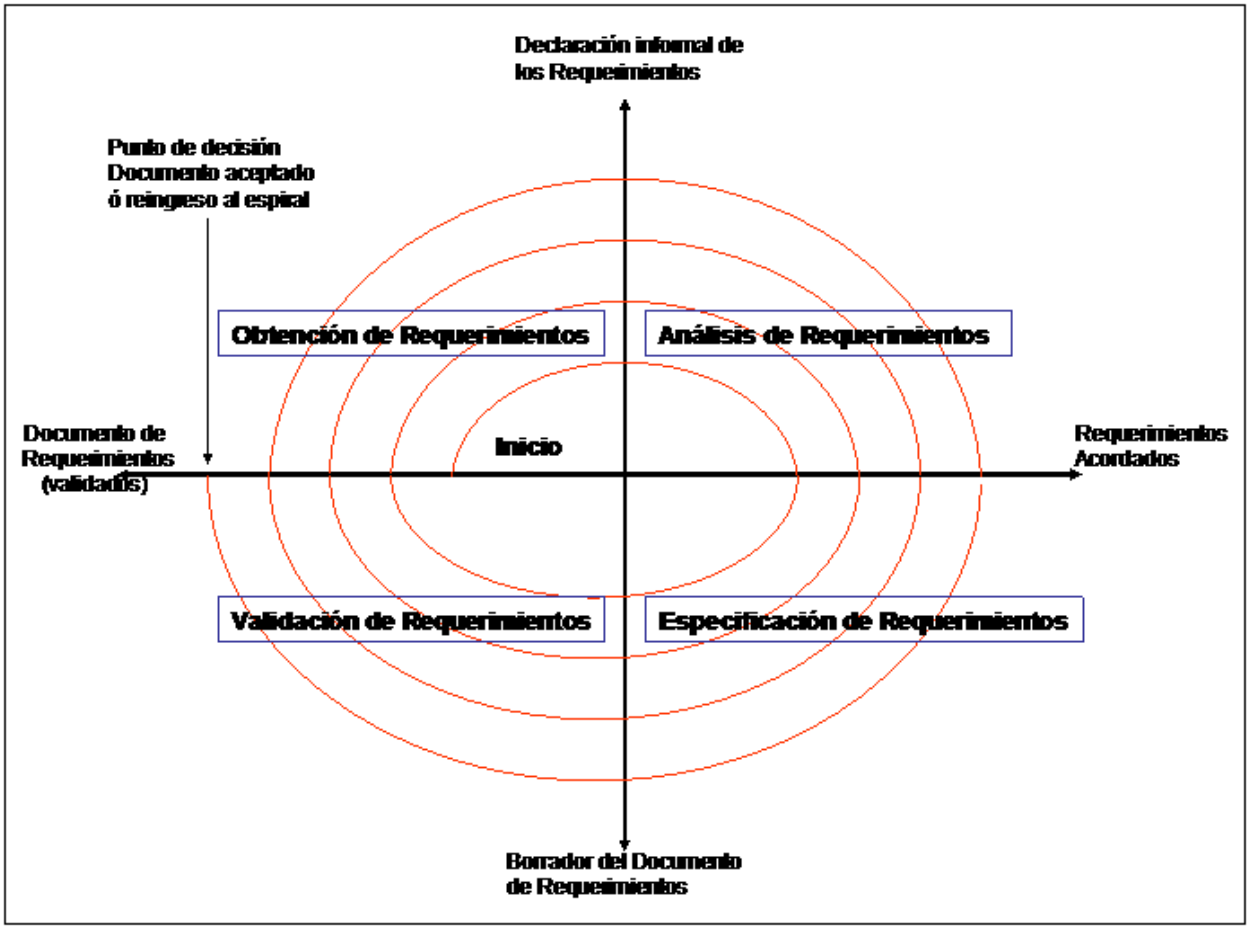


Figura 3.2 El modelo espiral del proceso de Ingeniería de Requerimientos.

Ambos modelos son adecuados para lograr el Documento de Especificación de los Requerimientos del sistema. Los dos posee cuatro actividades enlazadas que permiten avanzar o retroceder en el proceso en cualquier momento del desarrollo, esto permite lograr un documento de requerimientos completo y consistente.

3.3 Actividades del Proceso de Ingeniería de Requerimientos

El objetivo de la Ingeniería de Requerimientos es obtener un documento formal donde se especifiquen las características que debe cumplir el sistema que se va a desarrollar. Estas características y restricciones son definidas en común acuerdo por el cliente y los usuarios del sistema. La especificación de los requerimientos debe cumplir con ciertas características de calidad, como las siguientes: entendibles, necesarios, consistentes, no ambiguos, completos, verificables, correctos, reales, rastreables y modificables.

En cada una de las actividades del proceso de la Ingeniera de Requerimientos se llevan a cabo tareas específicas utilizando técnicas de requerimientos, modelos de especificación y herramientas para la administración del documento de requerimientos.

3.3.1 Obtención de Requerimientos

En esta actividad se determina el dominio de la aplicación, se especifican los servicios que deben proveer el sistema, la funcionalidad requerida del sistema, y las restricciones de hardware y software. Es indispensable la participación de los usuarios y clientes para la identificación de los requerimientos del sistema.

Se debe obtener como resultado de esta actividad, un documento de definición de los requerimientos, donde se definen las necesidades inicialmente encontradas, esto no significa que estos requerimientos sean los definitivos, pueden ser agregados nuevos requerimientos conforme se vayan encontrando o incluso los requerimientos establecidos pueden modificarse o eliminarse.

Para lograr la obtención de los requerimientos se definieron tareas a seguir y son las siguientes.

1. **Comprender el problema que se esta resolviendo**, es necesario estudiar el dominio o entorno en el que el sistema va a operar.
2. **Buscar y recolectar información** de manuales de operación y mantenimiento, de manuales organizacionales y políticas de operación.
3. **Definir los límites y restricciones del sistema**, para determinar con exactitud que es lo que el sistema va hacer y también especificar lo que no va hacer.

4. **Identificar a las personas o usuarios interesados por el sistema**, ya que ellos conocen el medio ambiente en que operará el sistema y pueden ayudar describiendo sus necesidades.
5. **Recolectar y clasificar requerimientos**, de esta manera los desarrolladores pueden iniciar definiendo un bosquejo del sistema, su funcionamiento básico y estableciendo el alcance del sistema

El desarrollo de estas tareas en la obtención de requerimientos es realizado secuencialmente, pero es necesario que cada tarea puede regresarnos a la anterior, sobretodo si no se descubre la información necesaria en el primer recorrido del diagrama. La figura 3.3 muestra esta relación. La salida de esta actividad nos conduce hacia el análisis de requerimientos.

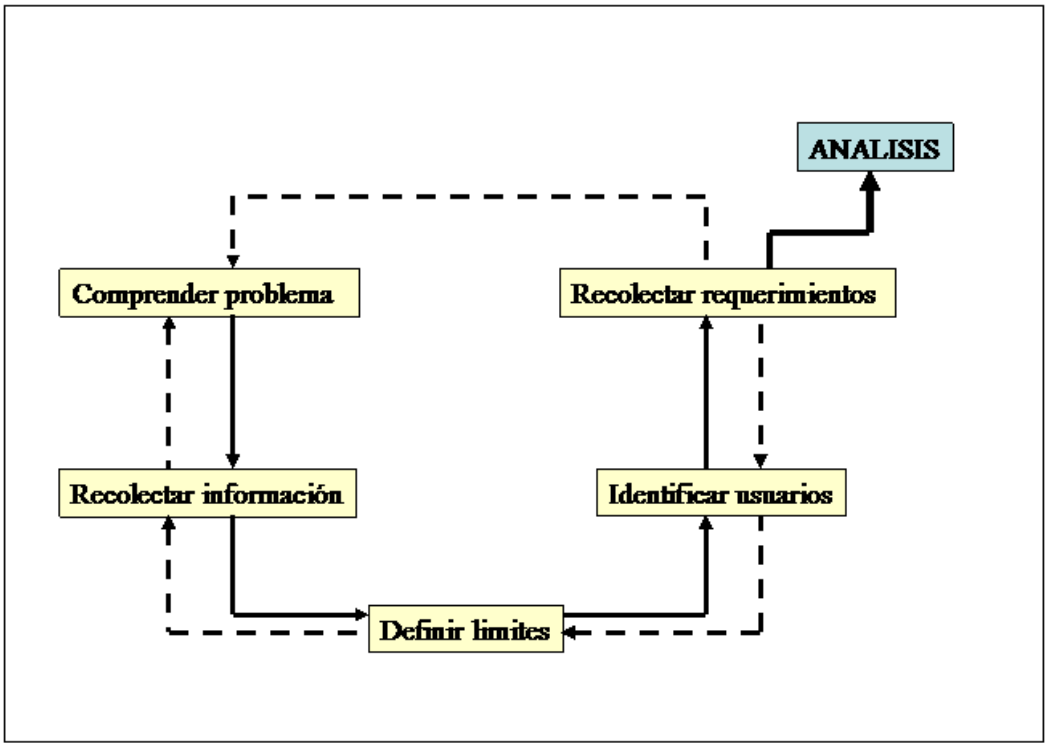


Figura 3.3 El proceso de obtención de requerimientos.

3.3.2 Análisis de Requerimientos

Los requerimientos definidos en el documento de definición son analizados detalladamente por el equipo de desarrollo y negociados con el cliente y usuarios, para decidir los requerimientos que serán aceptados y definirlos de manera conjunta con el fin de homogenizar su interpretación.

El análisis del **documento de definición de requerimientos** se lleva a cabo mediante técnicas de revisión y verificación de los criterios de calidad de cada requerimiento definido, este estudio es realizado por los desarrolladores, clientes y usuarios.

En el análisis se llevan a cabo las siguientes actividades:

1. **Priorizar los requerimientos** debido a que pueden existir requerimientos más importantes que otros para los clientes y usuarios, por lo que deben ser clasificados e implementados de acuerdo a su prioridad en el sistema.
2. **Encontrar dependencias entre requerimientos y etiquetar los requerimientos** con un identificador único, con el fin de poderlos identificar o rastrear en el futuro.
3. **Resolver conflictos entre los requerimientos**, se pueden encontrar conflictos entre requerimientos mediante la revisión de los criterios de calidad que debe cumplir cada requerimiento del sistema.
4. **Negociar con flexibilidad** con los demás elementos del equipo que intervienen en el proceso de desarrollo de software, para homogenizar su comprensión y de esta forma tanto desarrolladores como usuarios tengan la misma interpretación al momento de leer el documento de requerimientos.

El resultado del análisis es el **Documento de Definición de Requerimientos** (DDR), donde se encuentran descritos los requerimientos iniciales del sistema que se va a desarrollar en lenguaje natural; este documento sirve como punto de partida hacia la siguiente actividad del proceso de la Ingeniería de Requerimientos, por lo que es necesario de que no existan requerimientos en conflictos y estén bien escritos. La figura 3.4 describe este proceso, muestra la interacción de las tareas realizadas y la salida de esta actividad es la entrada a la actividad de especificación de los requerimientos.

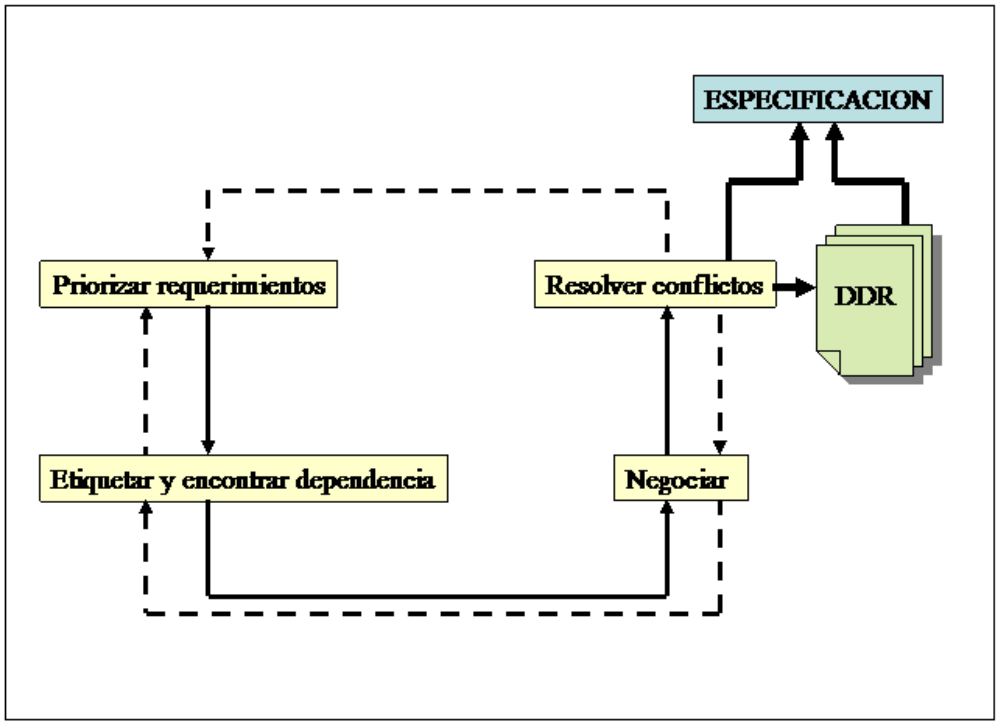


Figura 3.4 El proceso de análisis de requerimientos.

3.3.3 Especificación de Requerimientos

En esta actividad se obtiene como resultado el **documento de especificación de requerimientos**. Este documento contiene la descripción precisa de los requerimientos, incluye modelos de representación que agregan detalles al sistema mostrándolo desde diferentes perspectivas.

Para el desarrollo de esta actividad se realizan las siguientes tareas:

1. **Especificar los requerimientos funcionales y no funcionales**, ambos requerimientos deben ser descritos a detalles para su mejor comprensión.
2. **Determinar el tipo de estándar** a utilizar para la definición del Documento de Especificación de Requerimientos (DER). Se define el esquema del contenido del DER en base al estándar definido por la IEEE.
3. **Elegir la herramienta de especificación**, en caso de utilizar alguna para automatizar el proceso.

4. **Utilizar modelos y diagramas** para explicar con mayor detalle y diferentes perspectivas el comportamiento del sistema.
5. **Utilizar cualquier información de soporte o guía** para etapas posteriores y que amplíen la comprensión del sistema.

Todas estas tareas van encaminadas a obtener el Documento de Especificación de Requerimientos, en donde se especifican las funciones, restricciones o características del sistema en desarrollo. Es necesario contar con toda la información disponible para formar un documento completo y que sirva como base de partida hacia las otras etapas del desarrollo del sistema. En la figura 3.5 se muestran las actividades de la especificación de los requerimientos del sistema.

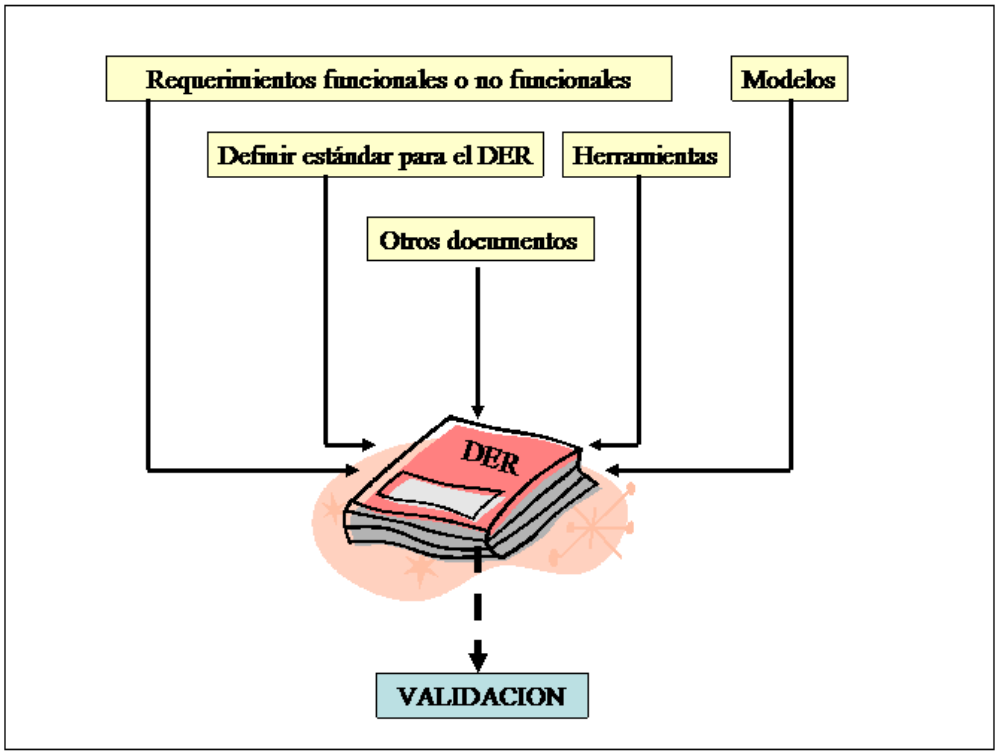


Figura 3.5 Las actividades de la especificación de requerimientos.

3.3.4 Validación de Requerimientos

La validación permite detectar problemas en el documento de requerimientos, y se lleva a cabo verificando cuidadosamente la consistencia y completitud del DER, antes de que sea usado como base en etapas posteriores del proceso de desarrollo del sistema.

La validación es un proceso importante debido a que permite detectar errores en el documento de requerimientos, de forma que puedan ser corregidos a tiempo. Estos errores si no son descubiertos en esta actividad pueden conducir a costos excesivos, ya que producen que se repita el trabajo cuando los errores sean descubiertos en etapas posteriores del desarrollo del sistema o en caso drásticos, si llegan a ser detectados cuando el sistema este en servicio.

Los criterios que son revisados para validar el documento de especificación de requerimientos son los siguientes:

1. **Verificar validez**. Los requerimientos satisfacen a las necesidades de los usuarios y son necesarios.
2. **Verificar consistencia**. Los requerimientos en el documento no deben contradecirse, ya que esto generaría conflictos en el momento de implementarse.
3. **Verificar integridad**, es decir, deben estar todos los requerimientos que describan todas las funciones y las restricciones propuestas por el usuario del sistema.
4. **Verificar realismo**, asegurar que los requerimientos puedan implementarse con la tecnología existente, considerando también el presupuesto y la calendarización del desarrollo del sistema.
5. **Generar casos de prueba** para los requerimientos.

Todos estos criterios deben ser orientados hacia la aplicación en el Documento de Especificación de Requerimientos, a fin de lograr su certificación y validación. Esta actividad es importante y debe tomarse en cuenta en cualquier proceso de desarrollo de software porque garantiza que lo expresado en el documento detallada las características del sistema que se va a desarrollar y puede servir de base para las etapas de diseño, implementación y pruebas del software. En la figura 3.6 se muestra esta actividad.

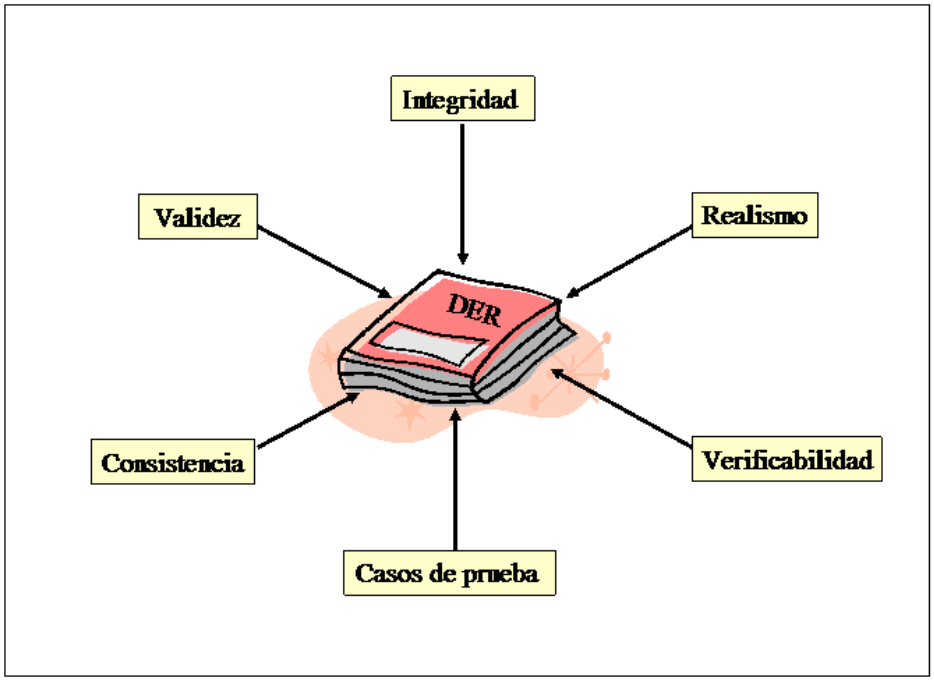


Figura 3.6 Las actividades de la validación del DER.

3.3.5 Administración de Requerimientos

La administración de los requerimientos describe el proceso de comprender y controlar los cambios en los requerimientos del sistema, Esta actividad comienza en cuanto existe la primera versión del documento de requerimientos ó cuando se planea darle mantenimiento a un sistema existente. Específicamente, en esta actividad se requiere lo siguiente:

1. **Administrar los cambios de los requerimientos**. Esto implica el análisis del problema y su propuesta de cambio, valorar el costo del cambio propuesto y finalmente implementar los cambios en los documentos que lo requieran.
2. **Establecer políticas de rastreo**. Es necesario conocer el origen de los requerimientos o los efectos que tendrán los cambios en los requerimientos relacionados.
3. **Control de versiones del documento de requerimientos** cuando implique algún cambio para referencias futuras.

3.4 Tipos de Requerimientos

Los requerimientos de software son las descripciones de los servicios y restricciones del sistema en desarrollo. Existen diferentes maneras de clasificarlos, generalmente se le agrupan de acuerdo a la audiencia a quienes van dirigidos y a las características del sistema.

3.4.1 Tipos de Requerimientos de acuerdo a la audiencia

Existen diferentes niveles de descripción de requerimientos, que permiten orientar los requerimientos a diversos usuarios. Es distinto explicar los alcances de un sistema nuevo a clientes usuarios que ni siquiera van a utilizar el sistema, pero que de alguna forma están involucrados en el desarrollo del sistema, que explicárselos a los desarrolladores o arquitectos de software que se encargaran de la implementación del sistema. Como se observa, se requiere de diferentes niveles de descripción y por lo tanto de diferentes tipos de descripción de requerimientos:

1. **Los Requerimientos del Usuario**. Son expresados en lenguaje natural utilizando diagramas fáciles de comprender, de los servicios que se espera que el sistema provea y de las restricciones bajo las cuales el sistema debe funcionar, también son conocidos como *Requerimientos del Cliente* o *Requerimientos C*.
2. **Los Requerimientos del Sistema**. En estos requerimientos se establecen con más detalle los servicios y restricciones del sistema. También se les conoce como *especificación funcional*, *Requerimientos del Desarrollador* o *Requerimientos D*.
3. **La Especificación del Diseño del software**. Es una descripción abstracta del diseño del software que se utiliza como una base para un diseño e implementación detallados.

Los distintos niveles de descripción de requerimientos son necesarios debido a que informan de manera clara a diferentes tipos de usuarios. Por ejemplo, los **requerimientos de usuario**, como están expresado en un lenguaje claro, fácil y sin detalles funcionales, están dirigidos a clientes administradores, usuarios finales del sistema, clientes ingenieros, contratistas administradores y arquitectos del sistema. Por otro lado, los **requerimientos del sistema** son dirigidos para usuarios técnicos del sistema, clientes ingenieros, arquitectos del sistema y desarrolladores de software.

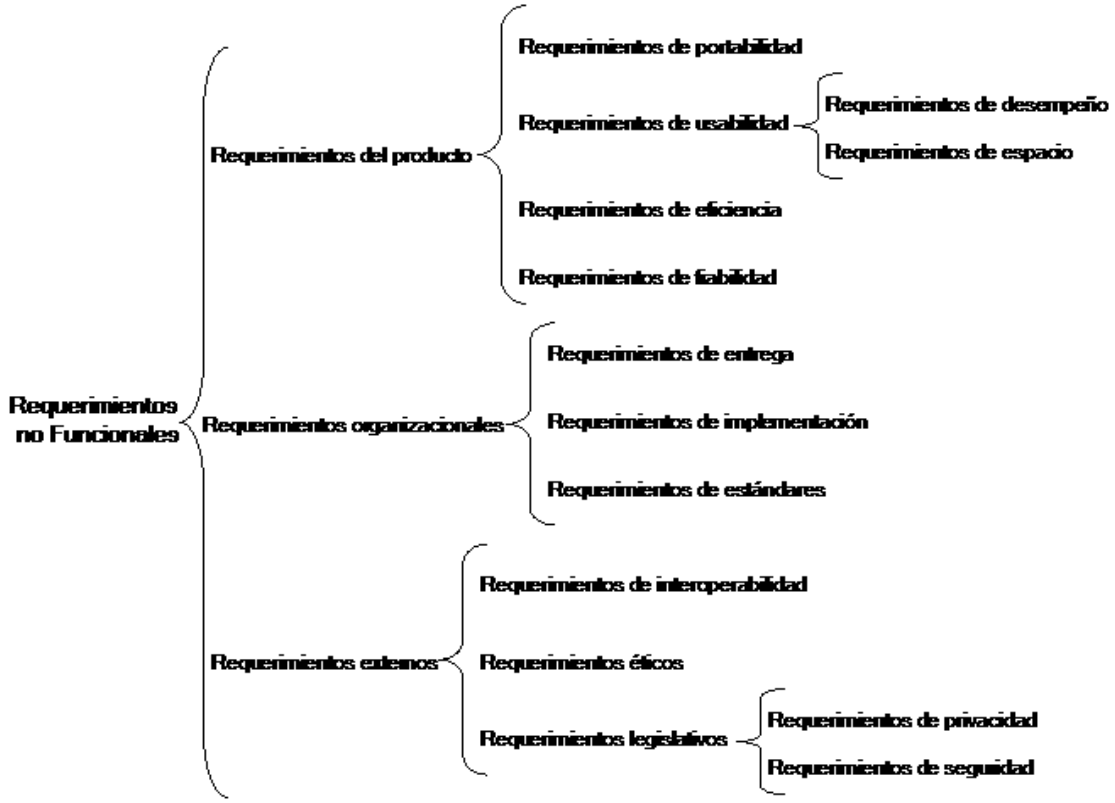
Debido a su especificación funcional dentro del sistema, es necesario de un cierto grado de conocimiento técnico ó especialización en el área. Y por último, la **especificación del diseño de software** está dirigida a clientes ingenieros, arquitectos del sistema y desarrolladores de software.

3.4.2 Tipos de Requerimientos de acuerdo a su característica

Esta clasificación se realiza en función de la naturaleza de las características del sistema que se desarrolla, generalmente los requerimientos de sistemas de software se clasifican en funcionales, no funcionales.

1. **Requerimientos funcionales**. Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema y comprenden la interacción entre el sistema y su ambiente, la reacción a entradas particulares de datos y su comportamiento en condiciones específicas. En algunos casos también declaran lo que el sistema no debe hacer.
2. **Requerimientos no funcionales**. Los requerimientos no funcionales, son especificaciones de las restricciones de los servicios o funciones ofrecidas por el sistema, restricciones sobre el sistema que limitan nuestras elecciones en la solución a un problema.

Los requerimientos no funcionales no se refieren directamente a las funciones específicas que entrega el sistema, sino a sus propiedades emergentes. Existen muchas maneras de clasificarlos, Sommerville [2] propone la siguiente clasificación.



- **Requerimientos del producto**, especifican el comportamiento del producto, por ejemplo el desempeño en la rapidez de ejecución del sistema, espacio de memoria requerida, la fiabilidad ó tasa de fallas para que el sistema sea aceptable, portabilidad y usabilidad.
- **Requerimientos Organizacionales**, se derivan de las políticas y procedimientos existentes en la organización del cliente y desarrollador. Son ejemplo de éstos los estándares que deben utilizarse en los procesos, lenguajes de programación en la implementación o métodos de diseño, los requerimientos que se especifican cuando se entrega el producto y documentación.
- **Requerimientos externos**, se refieren aquellos que se derivan de los factores externos al sistema y de su proceso de desarrollo. Por ejemplo, los requerimientos de interoperabilidad que establecen la manera en que el sistema interactúa con otros sistemas de la organización, requerimientos legales que aseguran que el sistema opere dentro de la ley y los éticos que aseguran que el sistema será aceptado por el usuario y el público en general.

3.4.3 Otros tipos de Requerimientos

Es muy difícil la clasificación de los requerimientos, pero lo importante en todo proceso tal vez no sólo sea la clasificación, sino la obtención de todos los requerimientos del software a desarrollar, su especificación y administración dentro de la etapa definida para este fin. Existen otras clasificaciones de los requerimientos en la bibliografía de Ingeniería de Requerimientos y que permiten definir de manera especifica las necesidades de los usuarios, estos tipos de requerimiento son los siguientes:

1. **Requerimientos de dominio.** Los requerimientos de dominio, son requerimientos que provienen del dominio de aplicación del sistema y reflejan las características de este dominio.
2. **Requerimientos de Datos.** Los requerimientos de datos definen las estructuras de datos requeridas en el sistema.
3. **Requerimiento de Interfaz.** Definen las características y parámetros de la comunicación del sistema a desarrollar con otros sistemas dentro de la empresa, o incluso de los subsistemas.

Los requerimientos evolucionan o tienden a cambiar de acuerdo a las necesidades del negocio o el desarrollo del software puede llevar tiempo sobre todo si se trata de sistemas grandes. Desde esta perspectiva los requerimientos se clasifican en:

1. **Requerimientos duraderos**. Estos tipos de requerimientos son relativamente estables debido a que se derivan de la actividad principal de la organización y porque están relacionados directamente con el dominio del sistema.
2. **Requerimientos volátiles**. Estos requerimientos sufrirán cambios probablemente durante el desarrollo del sistema o después de que este se haya puesto en operación.

Existe la siguiente subclasificación de los requerimientos volátiles:

1. **Requerimientos mutantes**. Son requerimientos que cambian debido a los cambios del ambiente en el que opera la organización.
2. **Requerimientos emergentes**. Son requerimientos que surgen al incrementarse la compresión del cliente en el desarrollo del sistema. El proceso de diseño puede producir nuevos requerimientos emergentes.

3. **Requerimientos consecutivos**. Estos requerimientos, son el resultado de la introducción del sistema. Esta introducción puede cambiar los procesos de la organización y abrir nuevas formas de trabajar que generarán nuevos requerimientos del sistema.
4. **Requerimientos de compatibilidad**. Son requerimientos que dependen de sistemas particulares o procesos de negocios dentro de la organización.

3.5 Características de calidad de los requerimientos

Los requerimientos no sólo describen el flujo de información de entrada y salida del sistema, y la transformación de los datos dentro del sistema, sino también las restricciones de rendimiento del sistema. De la misma manera, los requerimientos pueden servir a los siguientes propósitos:

- Permiten que los desarrolladores expliquen cómo han entendido lo que el cliente necesita del sistema.
- Indican a los diseñadores que funcionalidad y características va a tener el software resultante.
- Indican al equipo de prueba las demostraciones ó pruebas que realizaran para convencer al cliente de que el sistema que se le entrega es lo que había pedido.

La calidad de los requerimientos esta dada de acuerdo a las características que tengan y que permitan garantizar a los desarrolladores y clientes su entendimiento y utilización. Estas características son:

1. **Entendible.** Un requerimiento es entendible si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
2. **Necesario.** Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
3. **Consistente.** Un requerimiento es consistente si no es contradictorio con otro requerimiento.
4. **No ambiguo.** Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado para su especificación, no debe causar confusión al lector.

5. **Completo.** Un requerimiento esta completo si no necesita ampliar detalles en su redacción, por lo que se proporciona toda la información suficiente para su comprensión.
6. **Verificable.** Es verificable cuando puede ser cuantificado de manera que nos permita hacer uso de métodos de verificación (inspección, análisis, demostración o pruebas).
7. **Correctos.** Los requerimientos deben ser revisados por el cliente y desarrollador, para asegurar que no tienen errores.
8. **Reales.** Todos los requerimientos deben ser revisados para asegurar que son posibles de implementar en el sistema.
9. **Rastreables.** Los requerimientos pueden ser rastreables hacia atrás y hacia delante. Rastreable hacia atrás significa que para cada requerimiento se conoce su origen. Rastreable hacia delante significa que todo requerimiento esta escrito de tal forma que facilita la referencia a los requerimientos con que se relaciona.
10. **Modificable.** Un requerimiento es modificable si el lenguaje usado para su definición permite realizar cambios de manera fácil, completa y consistentemente.

3.6 Dificultades para definir los requerimientos de software

La Ingeniería de Requerimientos es un proceso muy complejo debido a la naturaleza y a la diversidad de ambientes de los sistemas, por lo que existe una gran variedad de situaciones que ilustran las dificultades que se enfrentan para lograr la obtención y especificación de los requerimientos del sistema. Estas dificultades son las siguientes.

1. Los usuarios o clientes del sistema no siempre pueden describir con exactitud lo que desean o necesitan.
2. Las descripciones de sus necesidades son expresadas utilizando sus propios términos y suposiciones con las cuales los desarrolladores pueden no estar familiarizados.
3. Diferentes usuarios tendrán distintas percepción de un mismo requerimiento, una visión contradictoria o con distinta prioridad.
4. Los usuarios conocen su trabajo y su sistema actual, pero no siempre pueden describir sus problemas a personas externas.

5. Los usuarios con frecuencia no conocen realmente lo que desean del sistema, y piden demandas irreales debido a que ignoran el costo de sus peticiones.
6. Los clientes encargados de hablar con los desarrolladores pueden no tener experiencia directa en hacer el trabajo que hacen los usuarios del sistema actual.
7. El equipo de desarrollo tiene que descubrir todas las fuentes potenciales de requerimientos, así como las partes comunes y en conflicto, debido a que diferentes usuarios tienen distintos requerimientos y podrían expresarlos de diferentes formas.
8. Los desarrolladores conocen soluciones al sistema, pero no siempre saben como afectará las soluciones posibles a las actividades del negocio de los clientes.
9. Los desarrolladores tienen también su propio lenguaje, y en la mayoría de las veces creen estar hablando en los mismos términos con el cliente, cuando en realidad dan significados diferentes a la misma cosa.
10. Si la comunicación no está cuidadosamente organizada y fomentada entre el equipo de desarrollo y los clientes, puede suceder que los requerimientos sean mal entendidos o queden incompletos.
11. El entorno económico y de negocio en el que se lleva a cabo el análisis es dinámico y la importancia de ciertos requerimientos puede cambiar.

3.7 El documento de especificación de Requerimientos

Este documento es conocido también como documento de Especificación de Requerimientos de software y es la declaración oficial de qué es lo que requieren los clientes del sistema. Incluye tanto los requerimientos funcionales como los requerimientos no funcionales del sistema. Alternativamente, la Definición de los Requerimientos del sistema se podrían definir en una introducción del Documento de Especificación de los Requerimientos del sistema y de esta forma integrar ambos documentos en un sólo documento. Si existen un gran número de requerimientos, los detalles de los requerimientos del sistema se podrían presentar como un documento separado.

El documento de requerimientos está dirigido a los usuarios y clientes quienes verifican que realmente estén definidos todos los servicios y restricciones que requieren del sistema, así también el documento de especificación de requerimiento sirve a el equipo de desarrolladores como una base para las siguientes fases del proceso de desarrollo, es decir, para elaborar el diseño y la implementación del sistema (figura 3.7).

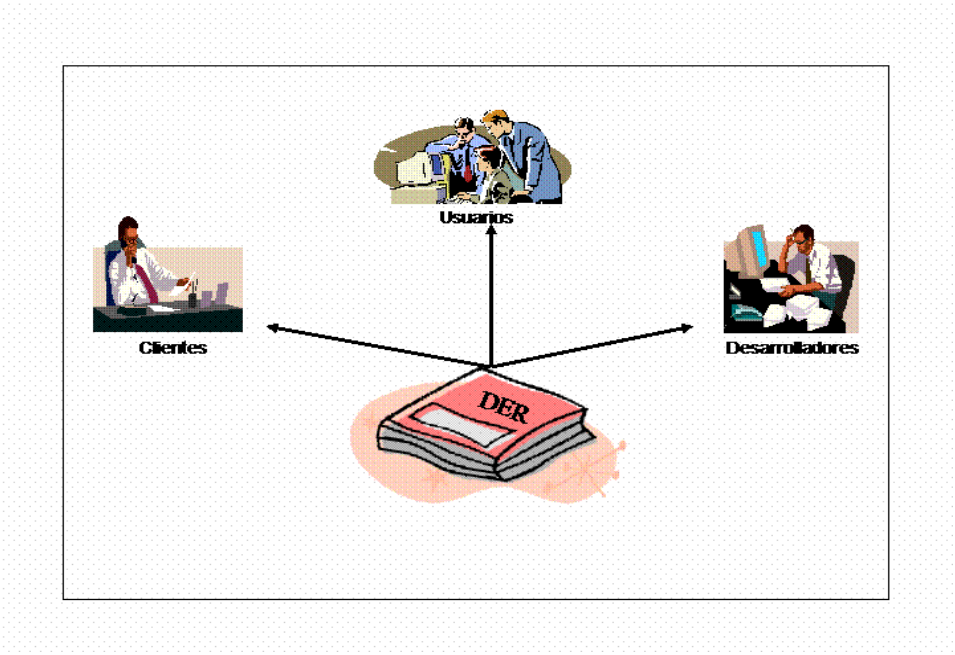


Figura 3.7 El DER es orientado a clientes, usuarios y desarrolladores.

Algunos de los requisitos que debe satisfacer un documento de requerimientos de software son listados a continuación:

1. Especificará únicamente el comportamiento externo del sistema.
2. Especificará las restricciones de la implementación.
3. Será fácil de cambiar.
4. Servirá como herramienta de referencia para los mantenedores del sistema.
5. Caracterizará las respuestas aceptables para los eventos no deseados.

3.8 Estándares de documentación

El Documento de Especificación deberá estar escrito de una manera correcta y formal de tal modo que tenga solo una interpretación por las personas involucradas, para esto se revisaron varios estándares y modelos de representación que fueron propuestos por gentes e instituciones expertas en la Ingeniería de Software. De esta forma, se analizaron todas las propuestas para obtener y presentar una manera de organizar el Documento de Especificación de Requerimientos del caso de estudio aquí propuesto.

3.8.1 El Estándar IEEE/ANSI 830-1993

Existen diferentes formas de definir y especificar los requerimientos en un documento formal, por lo que un gran número de diferentes organizaciones se han preocupado por definir estándares para los documentos de requerimientos. El más usual es el propuesto por la IEEE y la ANSI conocido como el estándar IEEE/ANSI 830-1993, también brevemente llamado como, el estándar 830-1993, que sugiere la siguiente estructura para los documentos de requerimientos:

1. **Introducción**
   - Propósito del documento de requerimientos
   - Alcance del producto
   - Definiciones, acrónimos y abreviaturas
   - Referencias
   - Resumen del resto del documento
2. **Descripción general**
   - Perspectiva del producto
   - Funciones del producto
   - Característica del usuario
   - Restricciones generales
   - Suposiciones y dependencias
3. **Requerimientos específicos**
   - Requerimientos funcionales
   - Requerimientos no funcionales
   - Requerimientos de interfaz.

   (Los requerimientos pueden documentar las interfaces externas, describir la funcionalidad y el desempeño del sistema, especificar los requerimientos lógicos de las bases de datos, las restricciones de diseño, las propiedades emergentes del sistema y las características de calidad)
4. **Apéndices**
5. **Índices**

El estándar propuesto es una guía o recomendación para la estructuración del documento de requerimientos. Sin embargo, se puede transformar y adaptar para definir un estándar que se ajuste a las necesidades de una organización en particular.

3.8.2 Otros Estándares para el Documento de Requerimientos

Otra manera de estructurar el documento de especificación de los requerimientos pero utilizando el estándar 830-1993 como base es el propuesto por Sommerville [2], que presenta el siguiente esquema:

1. Prefacio
2. Introducción
3. Glosario
4. Definición de requerimientos del usuario (descripciones en lenguaje natural, notaciones graficas y otras notaciones)
5. Arquitectura del sistema
6. Especificaciones de Requerimientos del usuario (detallar los requerimientos funcionales, los no funcionales, definir las interfaces con otros sistemas)
7. Modelos del sistema (mostrar las relaciones entre componentes)
8. Evolución del sistema
9. Apéndices
10. Índice (alfabético, diagrama, funciones, etc.)

En la bibliografía de Leszek [8] utiliza un esquema que contiene los siguientes puntos:

1. Proyectos preliminares
    - Propósito y alcance del producto
    - Contexto del negocio
    - Stakeholders
    - Ideas para la solución
    - Revisión de documentos
2. Servicios del sistema
    - Alcance del sistema
    - Requerimientos funcionales
    - Requerimientos de datos
3. Restricciones del sistema
    - Requerimiento de interfaz
    - Requerimientos de rendimiento
    - Requerimiento de seguridad
    - Requerimiento de operación
    - Requerimiento legales o de políticas
    - Otras restricciones
4. Materia del proyecto
    - Temas abiertos
    - Esquema preliminar
    - Presupuesto preliminar
5. Apéndices
    - Glosario
    - Documentos y formas del negocio
    - Referencias

Bernd [9] propone el siguiente esquema orientado al paradigma de objetos, el esquema tiene la siguiente estructura:

1. Introducción
    - Propósito del sistema
    - Alcance del sistema
    - Objetivos y criterios de éxito del proyecto
    - Definiciones, siglas y abreviaturas
    - Referencias
    - Panoramas
2. Sistema actual
3. Sistema Propuesto
    - Panorama
    - Requerimientos funcionales
    - Requerimientos no funcionales
        - Interfaz de usuario y factores humanos
        - Documentación
        - Consideración de hardware
        - Características de desempeño
        - Manejo de errores y condiciones extremas
        - Cuestiones de calidad
        - Modificaciones al sistema
        - Ambiente físico
        - Cuestiones de seguridad
        - Cuestiones de recursos
    - Pseudorequerimientos
    - Modelos del sistema
        - Escenarios
        - Modelos de casos de uso
        - Diccionarios de datos
        - Diagramas de clases
        - Modelos dinámicos
        - Interfaz de usuario
4. Glosario

Pressman [3] define un esquema donde describe las características del sistema usando diferentes modelos de representación como son los diagramas de flujos y de datos. En general, el esquema tiene la siguiente disposición:

1. Introducción
    - Metas
    - Objetivos
    - Contexto del sistema
    - Alcance del sistema
2. Descripción de la información
    - Descripción del problema
    - Relaciones, flujos y estructura
    - Interfaz de hardware, software y humana
3. Descripción funcional
    - Describir cada proceso (funciones)
    - Establecer y justificar restricciones de diseño
    - Características de rendimiento
    - Estructura global del sistema
4. Descripción del comportamiento
    - Comportamiento operativo
5. Criterios de validación

Bibliografía
Apéndices

Para presentar el documento de requerimientos del SIV, se propone la siguiente estructura que es una definición de los puntos más importantes de todos los enfoques estudiados y tomando como referencia el estándar IEEE/ANSI 830-1993.

1. Introducción
    - Propósito del documento de Especificación de Requerimientos del SIV
    - Alcance del documento de Especificación de Requerimientos del SIV
2. Descripción General
    - Descripción del sistema actual
    - Propósito y alcance del SIV
    - Contexto del SIV
    - Identificación de interesados (stakeholders)
3. Definición de los Requerimientos
    - Definición de los Requerimientos básicos del SIV.
    - Definición de los Requerimientos Funcionales del SIV usando lenguaje natural
    - Definición de los Requerimientos No Funcionales del SIV usando lenguaje natural
4. Especificación de los requerimientos del SIV
    - Especificación de Requerimientos Funcionales del SIV usando formas en lenguaje natural estructurado.
    - Modelos del sistema
        - Modelo de Datos
        - Modelos de Comportamiento
        - Modelos Orientados a Objetos
5. Criterios de Validación del Documento de Especificación de Requerimientos del SIV
6. Evolución del SIV
7. Apéndices
8. Glosarios

Capítulo 4

Técnicas para el proceso de Ingeniería de Requerimientos

En este capítulo se presenta una revisión de las técnicas más utilizadas en las actividades del proceso de la Ingeniería de Requerimientos. Se describen las técnicas de obtención de los requerimientos, donde el equipo de desarrollo interactúa continuamente con los usuarios del sistema para descubrir sus necesidades. En la actividad de análisis de requerimientos las técnicas son de para resolver conflictos entre requerimientos, negociar y establecer una definición común de los requerimientos establecidos en el Documento de Especificación de Requerimientos del sistema. En la especificación y documentación de los requerimientos las técnicas son necesarias para describir los requerimientos de manera que sean entendibles tanto para los usuarios como para los diseñadores. Además de las técnicas es necesaria la utilización de modelos para representar al sistema desde diferentes perspectivas, a fin de proporcionar una visión completa del Documento de Especificación. La validación del Documento de Especificación de Requerimientos consiste en que un grupo de usuarios y desarrolladores realicen una revisión del documento de requerimientos mediante técnicas que permitan certificar que el documento de requerimientos es consistente, completo, no ambiguo, rastreable, correcto y verificable. En la actividad de administración, se describen técnicas para el control de cambios que puedan surgir en los requerimientos y de versiones del documento de requerimientos. Finalmente se describen algunas de las metodologías más usadas en el proceso de la Ingeniería de Requerimientos.

4.1 Técnicas de Obtención de Requerimientos

Las técnicas de obtención de requerimientos son aquellas que nos permiten comprender el dominio del sistema, buscar y recolectar información para definir sus límites y restricciones e identificar a las personas interesadas y usuarios involucrados. El resultado permite obtener una colección y clasificación de los requerimientos del sistema, mediante la participación de los clientes y usuarios.

4.1.1 Entrevista

La entrevista es una técnica para obtener información mediante una conversación dirigida por los analistas a clientes y usuarios, con el objetivo de entender el dominio del problema y sus necesidades. Se basa en un formato de preguntas y respuestas. El desarrollador buscará obtener las opiniones de los clientes o usuarios entrevistados, sus opiniones del sistema, sus metas personales, de la organización y de los procedimientos informales.

Existen cinco puntos que deben tomarse en cuanta para preparar una entrevista [10]

a. **Lectura de antecedentes**. Se debe hacer un previo estudio de antecedentes de los entrevistados y su ambiente de trabajo, con ello se logrará conocer el lenguaje empleado dentro de la empresa.

b. **Establecer objetivos de la entrevista**. Se establecen los objetivos de la entrevista en base a los antecedentes y la experiencia personal.

c. **Selección de los entrevistados**. Se entrevista a gente clave de todos los niveles del sistema.

d. **Preparación del entrevistado**. Se debe de avisar a la persona que se entrevistará con tiempo para que pueda reflexionar acerca de la entrevista.

e. **Selección del tipo y la estructura de las preguntas**. Escribir las preguntas que cubran los aspectos fundamentales del objetivo de la entrevista, eligiendo el tipo y la estructura adecuada de las preguntas de la entrevista.

Las preguntas tienen ciertas estructuras básicas que se deben conocer. Los dos tipos básicos de preguntas son abiertas y cerradas.

1. **Las preguntas abiertas** permiten al entrevistado expresar de manera flexible y en consideración de su experiencia responder a lo que se le pregunta.
2. **Las preguntas cerradas** limitan las respuestas disponibles de los entrevistados mediante una lista de opciones o alternativas de respuestas.

La entrevista puede estructurarse de las siguientes maneras:

- **Estructura de pirámide**. En este caso, el entrevistador inicia con preguntas detalladas, del tipo de preguntas cerradas, y luego va haciendo preguntas abiertas, de respuestas más generalizadas.
- **Estructura de embudo**. El entrevistador comienza haciendo preguntas abiertas de carácter general y más adelante va utilizando preguntas cerradas.
- **Estructura de diamante**. Combina las estructuras anteriores, el entrevistador comienza de una manera muy específica, luego examina aspectos muy generales y finalmente llega a una conclusión muy específica.

Las entrevistas deben registrarse ya sea por un cuaderno de notas o utilizando una grabación. Es importante que al final de la entrevista se escriba un informe, con el fin de asegurar la calidad de los datos de la entrevista.

4.1.2 Cuestionarios

Es una técnica la cual constituye una manera de obtener información que permite a los desarrolladores del sistema recopilar opiniones, posturas, conductas y características de los diversos usuarios que son encuestados y que se encuentran involucrados en la operación del sistema actual o en la implantación de un sistema nuevo. Los cuestionarios son eficaces si la gente de la organización se encuentra dispersa o si esta involucrada mucha gente en el proyecto de sistema [10]. Al igual que la entrevista, un cuestionario puede redactarse usando preguntas abiertas o preguntas cerradas, además es importante usar un vocabulario que refleje el lenguaje de los involucrados en la organización. El uso de preguntas cerradas en los cuestionarios permite

a los analistas medir los resultados mediante gráficas. Esta técnica suele combinarse con las entrevistas para mejorar la obtención de información del sistema.

4.1.3 Puntos de Vista

En cualquier sistema existen varios tipos de usuarios que tienen diferentes intereses en los requerimientos del sistema, es decir, existen diferentes puntos de vista para un problema. Los enfoques orientados a puntos de vista toman en cuenta estos diferentes puntos de vista y los utilizan para organizar y estructurar tanto el proceso de obtención como los requerimientos [2]. Se toma en cuenta la existencia de diferentes perspectivas y proporciona un esquema para descubrir problemas con los requerimientos propuestos por diferentes usuarios.

Los puntos de vista pueden ser considerados como:

- **Una fuente o consumidor de datos**, son responsables de producir o consumir datos.
- **Un marco de trabajo de la representación**, se considera un tipo particular del modelo del sistema.
- **Un receptor de servicios**, son externos al sistema y reciben servicios de él.

Las lluvias de ideas y los puntos de vista orientados a eventos son técnicas que utilizan este enfoque.

a) Lluvias de ideas

En esta técnica se identifican los servicios potenciales y las entidades que interactúan con el sistema. Se lleva a cabo mediante una reunión entre el equipo de desarrollo y la empresa que solicita el software. Las personas involucradas generalmente son los usuarios del sistema y los administradores. En esta reunión todos expresan sus ideas sobre el problema y la posible solución. Cada persona participa sin ser interrumpida por otra, las ideas son anotadas dentro de un diagrama de burbuja en un pizarrón, y no pueden ser criticadas por los demás participantes. La figura 4.1 demuestra como son capturadas las ideas dentro de los diagramas de burbujas del sistema SIV. Al finalizar la sesión, se realiza una recolección de ideas sin duplicidad [2].

Figura 4.1 Lluvia de ideas para el sistema SIV.

4.1.4 Observación

Es una técnica en donde el desarrollador se sumerge en el ambiente de trabajo de los usuarios, para observar el trabajo diario que éstos realizan anotando los aspectos importantes, observando factores sociales y organizacionales que afecten el sistema. Existen dos enfoques de esta técnica y son la etnografía y grabar lo observado [11].

a) Etnografía

Generalmente los sistemas de información se encuentran dentro de un contexto social y organizacional, donde los requerimientos del sistema se derivan y se restringen conforme a este contexto. La etnografía es una técnica que se utiliza para entender estos tipos de requerimientos sociales y organizacionales. El desarrollador se involucra en el entorno laboral donde se utilizará el sistema, para observar y anotar las tareas reales que se llevan a cabo. Esta técnica descubre

requerimientos implícitos que reflejan los procesos reales más que los formales, en los que la gente está involucrada, y permite descubrir los factores sociales y organizacionales que puedan afectar al sistema. Una desventaja de este enfoque es que no siempre puede identificar nuevas propiedades o agregar otras al sistema [2].

4.1.5 Escenarios

Los escenarios son representaciones de como el usuario interactúa con el sistema. En este esquema se representan gráficamente las entradas, salidas, el flujo de datos y el comportamiento del sistema. Los escenarios pueden agregar detalles a un bosquejo de acción. Los escenarios son ejemplos de secciones de interacción de una simple operación entre el usuario y el sistema, el proceso de desarrollo de un escenario puede ayudar a entender los requerimientos [11]. Los escenarios son parte básica de algunos métodos de análisis orientados a objetos. Existen dos esquemas para esta representación: utilizando escenarios de eventos ó casos de uso.

a) Escenarios de eventos

Este enfoque se utiliza para documentar el comportamiento del sistema ante eventos específicos. Los escenarios de eventos incluyen una descripción del flujo de datos y las acciones del sistema, documenta las excepciones que pueden ocurrir. Un escenario de evento es una instancia de un caso de uso, es decir, nos representa una secuencia de sucesos que podrían ocurrir en una situación dada [2]. En la figura 4.2 se representa un diagrama de escenario de eventos para un cajero automático, se pide el número de identificación personal del cliente (PIN), se introduce la tarjeta y el PIN. Si la tarjeta es válida y el cajero puede procesar, entonces el control pasa a la siguiente etapa.

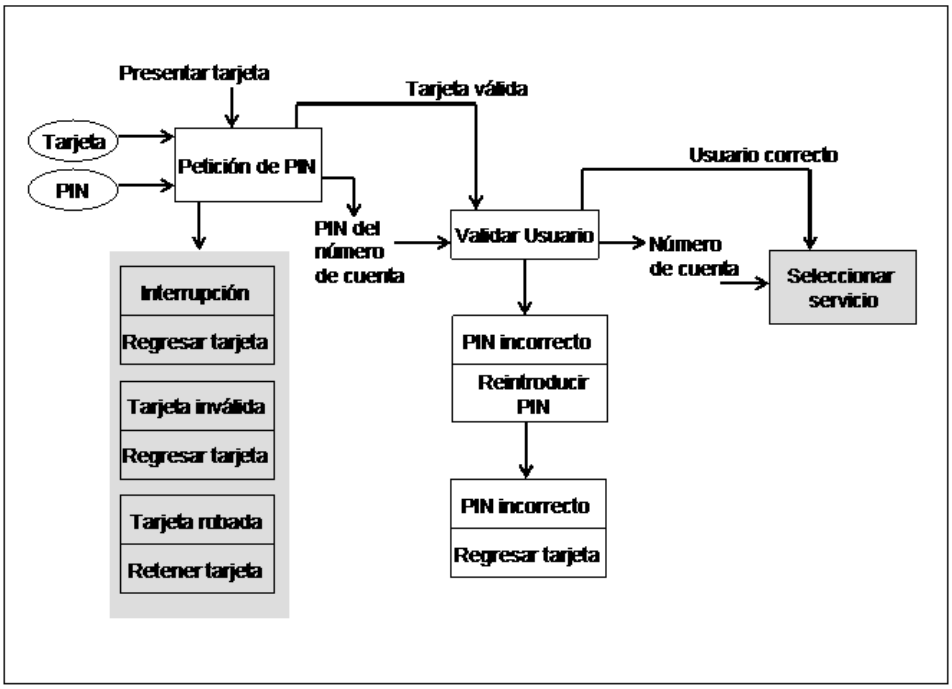


Figura 4.2 Escenario de eventos para transacciones de un cajero automático.

b) Casos de uso

Los casos de uso es una técnica que se basa en escenarios para la obtención de requerimientos [2]. En los casos de uso se especifica una secuencia de acciones, incluyendo variantes que el sistema puede llevar a cabo y que producen resultados observables de valor para un actor determinado [5]. En un modelo de caso de uso se puede identificar a un conjunto de actores involucrados y su relación con varios casos de uso. Un conjunto de casos de uso representan todas las posibles interacciones que ocurrirán en los requerimientos del sistema. Los actores pueden tener las siguientes representaciones: personas, sistemas o hardware y cualquier de las tres representaciones puede estar relacionada con un caso de uso.

Los requerimientos funcionales pueden estar representados mediante casos de uso y muchos de los requerimientos no funcionales pueden estar asociados a ellos. En la notación grafica de UML, se representa a un actor con un símbolo de muñeco, un caso de uso con una elipse y la relación

entre ellos mediante una línea que puede tener una punta de flecha indicando el sentido de la relación. La figura 4.3 representa un modelo de casos de uso para una biblioteca.

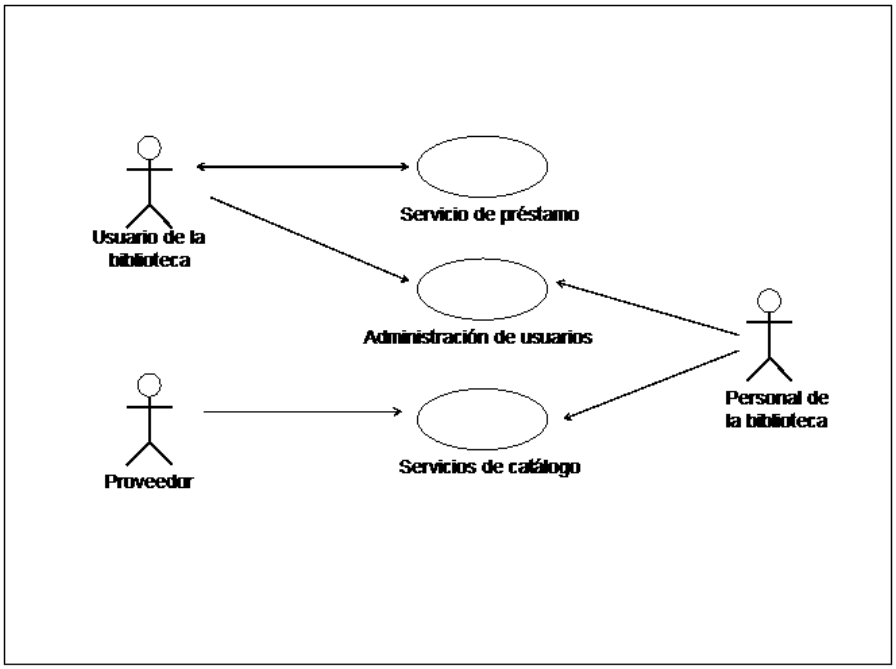


Figura 4.3 Representación de casos de uso para una biblioteca.

A pesar de la representación de diagrama en UML, algunos analistas describen los casos de uso de manera textual, anotando todos los detalles que ocurren en cada escenario. Para detallar más a los casos de uso, en UML se utilizan los diagramas de secuencia, diagramas de actividad, diagramas de colaboración y diagramas de estados.

4.1.6 Análisis de Protocolos

El análisis de protocolos es una técnica aplicada a un grupo de usuarios representativos, a quienes se les pide que describan en voz alta sus actividades dentro del sistema. Así el desarrollador podrá identificar las metas y submetas del negocio apreciadas por los usuarios.

4.1.7 Maestro/Aprendiz

Es una técnica en donde el papel del aprendiz es representado por los analistas o desarrolladores y los usuarios representan al maestro. Los desarrolladores, con el papel de aprendiz, participan en un entrenamiento con los usuarios finales del sistema. El aprendiz hace uso de la observación y podrá cuestionar sobre el origen y significado de las tareas que realice. También, el aprendiz podrá realizar algunos trabajos bajo la supervisión del maestro, con el fin de proponer bosquejos o alternativas de solución. El principal objetivo es proporcionar detalles al aprendiz en tareas específicas del usuario final [18].

4.1.8 Despliegue de la Función de Calidad (**Quality Function Deployment**)

Esta técnica fue propuesta por los japoneses para determinar requerimientos de calidad en la industria automotriz, se concentra en maximizar la satisfacción del cliente [3]. El enfoque principal de QFD es la ***Función de Calidad***, en la que se muestran las relaciones entre los requerimientos del cliente y las características del producto, es decir, mediante una matriz, en donde las filas representa los "que" y las columnas representan los "como". En otras palabras, tenemos los requerimientos que son los "que" y como se llevan a cabo esos requerimientos mediante casos de uso que vienen siendo los "como" (tabla 4.1).

Requerimientos	Caso de uso 1	Caso de uso 2	Caso de uso 3	Caso de uso n
R1	X		X	
R2		X		
R3	X	X		
…				
Rm		X		X

Tabla 4.1 Función de Calidad.

En la función de calidad se marca la intersección que exista entre un requerimiento y los casos de uso que lo implementan, verificando que todo requerimiento sea implementado por algún caso de uso y que todo caso de uso satisfaga algún requerimiento. Además se deben revisar que no

existan cosas repetidas (redactadas de diferentes maneras) o contradictorias tanto en la lista de requerimientos como en los casos de uso.

4.1.9 Talleres de Trabajo basados en Casos de Uso (**Run Use Case WorkShop**)

Los talleres de trabajo basados en casos de uso se realizan entre el cliente/usuario y el equipo de desarrolladores [13]. Esta técnica consiste en dos tareas principales:

1. **Generar los escenarios**, mediante la aplicación de los casos de uso se puede conversar con los usuarios para extraerles los detalles que suceden cuando realizan un evento determinado. De esta manera se podrán definir los actores que participan y definir los pasos que se realizan para el caso de uso en cuestión. Se debe verificar con los usuarios si los pasos están bien descritos, o pueden ser modificados y mejorados.
2. Al término de la etapa anterior el equipo de desarrolladores **describe y deduce los requerimientos** a partir del conocimiento previamente adquirido.

4.1.10 Análisis de Usuarios Interesados (**Stakeholder**)

Los ***stakeholders*** son las personas involucradas de alguna manera en el sistema y que permiten asegurar el éxito del proyecto, por ejemplo los usuarios finales y sus administradores, clientes y dueños del negocio, ingenieros de sistemas y programadores [12]. Es importante encontrar a todos los grupos de stakeholders y conocer sus intereses. Durante el análisis de stakeholders se debe tratar de encontrar respuestas a las siguientes preguntas [11]:

- ¿Quiénes son los stakeholders?
- ¿Qué objetivos visualizan para el sistema?
- ¿Cómo podrían contribuir en el sistema?
- ¿Qué riesgos y costos ven?
- ¿Qué tipo de soluciones observan, para el sistema?

Hay varias maneras de obtener esta información, puede obtenerse mediante una reunión grupal, donde todos los stakeholders conocidos estén representados ó pueden citarse grupos pequeños de stakeholders a reuniones. En caso de que esto fallara, puede entrevistarse a uno por uno del grupo de stakeholders identificados.

4.1.11 Demostración de Tareas

La técnica de demostración de tareas es una variante de la entrevista y la observación, y consiste en preguntar a los usuarios como realizan una tarea específica del sistema. Es útil cuando los usuarios no pueden explicar lo que hacen en su trabajo diario, pero si pueden demostrar como realizan tareas específicas. La demostración de tareas es también una manera poco frecuente de observar tareas críticas. A los usuarios se les hace más fácil demostrar como realizan sus tareas, que explicar con palabras como se desarrollan las cosas incluso antes de que sean introducidas al sistema actual. Otra aplicación de la demostración de tarea es la identificación de problemas de utilidad, para la identificación de problemas de utilidad en un sistema nuevo es necesario algún tipo de demostración de tareas [12].

4.1.12 Enfoque Grupal (**Focus Groups**)

Esta técnica consiste en reuniones grupales donde se fomenta la discusión abierta entre los participantes. La discusión proporciona al analista ideas de cómo los usuarios piensan y sienten acerca de los aspectos del sistema. Este enfoque puede ser usado en el desarrollo del proceso de la Ingeniería de Requerimientos, es decir, en la etapa inicial del proyecto, el enfoque grupal puede ser utilizado para obtener requerimientos basándose en lo que los usuarios piensan que el sistema podría abarcar y durante el desarrollo podría ser usado para evaluar prototipos y proporcionar validación y verificación del producto [19].

4.1.13 Talleres Futuros (**Future Workshops**)

Esta técnica es similar al enfoque grupal, pero es más estructurada y muy concreta en su alcance. Consiste en que los participantes definan el estado futuro deseado del ambiente del sistema. Los facilitadores se aseguran de que el alcance del sistema sea definido y respetado, además permite a los participantes discutir el estado final deseado del sistema [14].

4.1.14 Estudio de Documentos/Datos

Esta técnica consiste en estudiar los documentos existentes tales como formas, archivos de datos, bitácoras, documentación y base de datos del sistema existente. Se pueden analizar las pantallas desplegadas por el sistema en uso, si son impresas en papel. Otra función de esta técnica, es la de verificar la información recabada en las entrevistas realizadas a los usuarios y clientes del sistema. En general, lo que se desea lograr es determinar el contexto y la información detallada del sistema en la que los requerimientos están basados [15].

4.2 Técnicas de Análisis de Requerimientos

Los requerimientos deben ser analizados detalladamente tanto por el equipo de desarrollo como por los usuarios para resolver conflictos de ambigüedad y consistencia entre requerimientos, priorizando los requerimientos que han de ser inicialmente implementados en el sistema, además de negociarlos con flexibilidad para homogenizar su comprensión y de esta forma obtener una lista consistente de requerimientos. Para realizar el análisis se cuenta con diversas técnicas, como las que se mencionan a continuación.

4.2.1 La identificación de requerimientos

Se debe identificar a cada requerimiento de manera única, de tal forma que puedan permitir una referencia cruzada con otros requerimientos, y así utilizarse en las evaluaciones de rastreo. Para identificar cada requerimiento se pueden usar las letras iniciales del tipo de requerimiento y un número de referencia, por ejemplo si se tratara de un requerimiento funcional puede usarse la siguiente estructura RFn, donde n es un número entero consecutivo para cada requerimiento y para los requerimientos no funcionales RNFn. Ejemplos de identificación de requerimientos funcionales y no funcionales:

RF2.15 El coordinador podrá ver a todos los alumnos que se inscribieron a un curso en específico. *Es un requerimiento funcional del SIV*.

RNF8 El sistema de contar con interfaces gráficas de usuarios que utilicen los logotipos y colores institucionales. *Es un requerimiento no funcional del SIV*.

4.2.2 Listas de Verificación (**Checklist**)

Esta técnica es muy fácil de utilizar y proporciona una gran utilidad. En general, es una lista de preguntas que el analista debe usar para evaluar cada requerimiento. Los analistas deben verificar y marcar los puntos de esta lista mientras leen el documento de requerimientos, cuando se descubran problemas potenciales, deberán ser anotados, ya sea en los márgenes del documento, ó en una lista de verificación. Las *listas de verificación* son útiles porque brindan un recordatorio

de lo que se debe buscar y reducen la posibilidad de pasar por alto alguna verificación importante. No sólo son útiles para verificar requerimientos, sino también se puede aplicar con los casos de uso. El proceso de verificación se puede implementar con una hoja de cálculo en donde las filas representan, por ejemplo, los distintos requerimientos, y las columnas representan los puntos a verificar ó los criterios que deben cumplir, como se ilustra en la tabla 4.1. Entonces se completan las intersecciones que correspondan con los comentarios sobre los problemas potenciales.

ID	Entendible	Consistente	No ambiguo	Completo	Verificable	Correcto	Modificable
RF1	si	si	si	no	si	no	Si
RF1.1	si	si	si	si	si	si	Si
RF1.2	si	si	si	si	si	si	Si
RF2	si	Si	si	no	si	si	Si
RF2.1	no	Si	no	no	si	si	Si
RF2.2	no	Si	no	no	si	si	Si
RF2.3	no	Si	si	si	si	si	Si

Tabla 4.2 Lista de verificación de calidad en los requerimientos.

4.2.3 Matriz de dependencias entre Requerimientos

Suponiendo que los requerimientos sean claramente identificados y enumerados, entonces podemos utilizar una matriz de dependencia de requerimientos, donde se listen en orden todos los requerimientos identificados, como se muestra en la tabla 4.2.

Requerimientos	R1	R2	R3	R4
R1	X	X	X	X
R2	Conflicto	X	X	X
R3			X	X
R4		Intercalado	Intercalado	X

Tabla 4.3 Matriz de dependencia entre requerimientos.

Las celdas de la matriz son marcadas con una X si dos requerimientos tienen dependencia o se escribe en las celdas si están intercalados o en conflicto. Una celda vacía indica que los requerimientos son independientes. Los requerimientos en conflicto o intercalados podrán ser discutidos con los clientes para después ser replanteados y de este modo resolver los conflictos o la intercalación entre estos. La matriz de dependencia de requerimientos es una técnica simple pero efectiva para encontrar conflictos e intercalaciones cuando el número de requerimientos es relativamente pequeño. Sin embargo, cuando el número de requerimiento es mayor, la técnica aún puede ser usada si los requerimientos se agrupan en categorías, de esta forma los requerimientos pueden ser comparados separadamente en cada categoría.

4.2.4 Negociación de Requerimientos

El propósito de la negociación es resolver conflictos en el desarrollo del sistema. Los conflictos pueden encontrarse entre desarrolladores y clientes, pero los conflictos más serios surgen entre los grupos de stakeholders que existen dentro del ambiente del sistema. Los conflictos dentro del ambiente del sistema pueden presentarse de diferentes maneras. Un miembro de un grupo de discusión debe participar desde alguna posición dentro de las partes en problemas para poder entender el inconveniente y proponer una solución. Para lograr acuerdos, las partes en conflictos deben estar dispuestas a hablar e intentar entender la postura de la parte contraria. Si esto no sucede, el trabajo obtenido podría estar basado desde un punto de vista en particular y no de un común acuerdo de los participantes. De otra manera, pudiera ser resuelto por las decisiones de personas que se encuentran en niveles superiores en la organización. Existen varias maneras disponibles para resolver los conflictos, por ejemplo que cada parte en conflicto explique que le hace falta a la otra. Desde el punto de vista de los requerimientos, lo más importante es analizar los objetivos de las partes involucradas en el desarrollo del sistema. Generalmente, los conflictos son sobre las soluciones que se plantean, pero mediante la participación de todos los involucrados se pueden ajustar estas soluciones incluso los objetivos mismos. La meta es encontrar soluciones que no entren en conflictos, pero que impliquen todos los objetivos en común.

4.3 Técnicas de Especificación de Requerimientos

En esta actividad se obtiene el documento de especificación de requerimientos que es orientado a diversos tipos de lectores, por lo que es necesario contar con múltiples niveles de descripción. La especificación de los requerimientos debe incluir diferentes modelos para representar el sistema con mayor detalle y desde diversas perspectivas. Es necesario contar con modelos de representación, métodos, metodologías y herramientas que proporcionan la manera de describir los diferentes tipos de requerimientos del sistema en el *documento de especificación*. El documento es redactado para que sea interpretado por los usuarios y por los diseñadores del sistema, recordemos que los usuarios son personas sin conocimientos técnicos de implementación, por otro lado el equipo de diseño son expertos en el área, por lo que se requiere de ambas perspectivas del documento. Existen varias maneras de describir los requerimientos de software y se describen a continuación.

1. **Lenguaje Natural**
2. **Notación Gráfica**
3. **Especificación Matemáticas**

4.3.1 Lenguaje Natural

El lenguaje natural es utilizado en la mayoría de las veces para escribir los requerimientos del sistema con el fin de que los usuarios puedan comprenderlos sin necesidad de poseer conocimientos técnicos de desarrollo y comprobar que sus necesidades están expresadas en el documento de requerimientos. Por otro lado, aunque el lenguaje natural sea flexible y sencillo puede contener problemas de interpretación. Algunos problemas que se pueden encontrar en la descripción de los requerimientos utilizando el lenguaje natural son:

- **Falta de claridad**. Es difícil utilizar el lenguaje de forma precisa y no ambigua.
- **Confusión de Requerimientos**. Se dificultad la distinción de los requerimientos funcionales de los no funcionales, las metas del sistema y la información para el diseño.

- **Conjunción de Requerimientos**. Varios requerimientos diferentes se expresan de forma conjunta como un único requerimiento.
- **La comprensión del lenguaje natural** radica en que los lectores y redactores de la especificación utilicen la misma palabra para el concepto.
- **La especificación del lenguaje natural es demasiado flexible**. Se puede decir lo mismo de formas completamente diferentes.
- **No existe una forma fácil de modularizar los requerimientos**. El lenguaje natural es difícil exhibir todos los requerimientos relacionados.

Existen dos modificaciones del lenguaje natural que restringen la manera de redactar los requerimientos y son **lenguaje natural estructurado** y **lenguaje de descripción de diseño**.

a) Lenguaje Natural Estructurado

Este lenguaje es una forma restringida del lenguaje natural para expresar los requerimientos del sistema asegurando que se imponga un cierto grado de uniformidad a la especificación y manteniendo la expresividad y comprensión del lenguaje natural. Para la notación de este lenguaje es fundamental el uso de plantillas o formas estándar para especificar los requerimientos. Cada forma debe incluir la siguiente información:

1. Una descripción de la función o entidad a especificar.
2. Una descripción de sus entradas y de donde provienen.
3. Una descripción de sus salidas y hacia donde van.
4. Una descripción de que otras entidades se utilizan.
5. Si se utiliza un enfoque funcional, definir una precondición y poscondición.
6. Una descripción de los efectos colaterales (si existes) de la operación.

Con esto podemos lograr eliminar algunos problemas y obtener menos variación en la especificación, sin embargo pueden permanecer algunas ambigüedades en la especificación. La figura 4.4 muestra un ejemplo de una plantilla para especificar los requerimientos de software.

SIV: **RF1**

Función: Acceso al SIV

Descripción: El sistema permitirá dar acceso a usuarios, que serán identificados como: *Alumno* y *Coordinador* mediante un *login* y una contraseña, además de un usuario *público en general* que podrá usar el sistema sin necesidad de introducir un *login* y una contraseña.

Entradas: Login y contraseña del usuario.

Origen: Mediante el teclado de la computadora.

Salidas: Validación de usuario o mensaje de error.

Destino: Servidor del SIV (base de datos de accesos).

Requisitos: Servidor del SIV activo, bases de datos de accesos a Alumnos y coordinadores.

Pre-condiciones: Ejecución del servicio WWW en Internet del servidor SIV.

Post-condiciones: Validación de usuario.

Efectos laterales: Error al tratar de acceder al SIV sin estar registrado en la base de datos de usuarios.

Figura 4.4 Plantilla para especificar requerimientos de software en lenguaje natural estructurado.

b) Lenguaje de Descripción de Diseño

Los requerimientos pueden describirse de forma operacional utilizando un Lenguaje de Descripción de Programas (PDL), para anular las ambigüedades inherentes en la especificación en lenguaje natural. Un PDL contiene construcciones abstractas para incrementar su poder de expresividad, es decir, utiliza un lenguaje similar a un lenguaje de programación pero con características más genéricas. Su ventaja es que se verifica de manera sintáctica y semántica con herramientas de software. Las omisiones de los requerimientos y las inconsistencias se infieren de los resultados de las verificaciones. El uso de PDL nos lleva a la especificación detallada y algunas veces muy cercanas a la implementación. La figura 4.5 es un esquema de la especificación de requerimientos del SIV usando PDL.

```
Procedure  SIV is
            Load: DB_Alumnos;
            Load: DB_Personal;
            Login: LoginAcceso;
            Valid_Password: clave_acceso;
            Valid_Acces: Boolean;
  Begin:
            If (Valid_Acces == true)
              If (LoginAcceso == Personal_Admon)
                        Then
                                    Load: DB_Cursos;
                                    Load: DB_Alumnos;
                                    Get_service (Servicios);
                                    While(service is select) loop
                                                Deliver_servicie_selected;
                                                Get_service(Servicios);
                                    End_loop;
                        If (Change in DB_Cursos)
                           Then Actualiza:DB_Cursos;
                        If (Cahnge in DB_Alumnos)
                           Then Actualiza:DB_Alumnos;

            If  (LoginAcceso == Alumno)
                        Then
                                    Load: DB_Alumnos_Regular;
                                    Valid_Access: boolean;
                                                If (Valid_Access==true)
                                                            Load: DB_Cursos;
                                                            Get_service (Servicios);
                                                                        While(service is select) loop
                                                                                    Deliver_service_selected;
                                                                                    Get_service(Servicios);
                                                                        End_loop;
                        If (Change in DB_Cursos)
                           Then Actualiza:DB_Cursos;
     Close_All_DB: DB_Alumnos, DB_Cursos, DB_Personal;
Else (Valid_Access==False)
            Print (" Error access, El Usuario no es valido);
End_SIV.
```

Figura 4.5 Representación de requerimientos usando PDL.

En la figura 4.5, se indica la Especificación de Requerimientos usando PDL, se detalla lo que realiza el sistema SIV desde un punto de vista operacional completo.

4.3.2 Notaciones Gráficas

Es una manera ilustrativa de generar la especificación de los requerimientos mediante un conjunto de modelos. Los modelos utilizan un lenguaje gráfico, complementado con anotaciones textuales que describen el problema a resolver y el sistema a desarrollar. En muchos casos, los modelos de requerimientos del sistema son más entendibles que la descripción en lenguaje natural, debido a las representaciones gráficas. Son considerados vías directas entre los procesos de requerimientos y diseño.

Los modelos se utilizan en la Ingeniería de Requerimientos para desarrollar una comprensión del sistema existente a reemplazar, mejorar o para especificar el sistema requerido. Un modelo es una abstracción del sistema que se va a desarrollar, más que una representación alternativa de ese sistema, por lo que una abstracción omite detalles simplificando y seleccionando las características más sobresalientes de la entidad representada. En contraste, una representación de un sistema debe mantener toda la información. Por lo que existen diferentes tipos de modelos que se basan en varios enfoques de abstracción y representan al sistema desde diferentes perspectivas.

1. **Perspectiva Externa** la que se modela el contexto o entorno del sistema.

    a. *Modelos de Contexto*. Definen los límites del sistema, distinguiendo lo que es el sistema y su entorno, como se muestra en la figura 4.6 el modelo de contexto para un sistema de cajero automático.

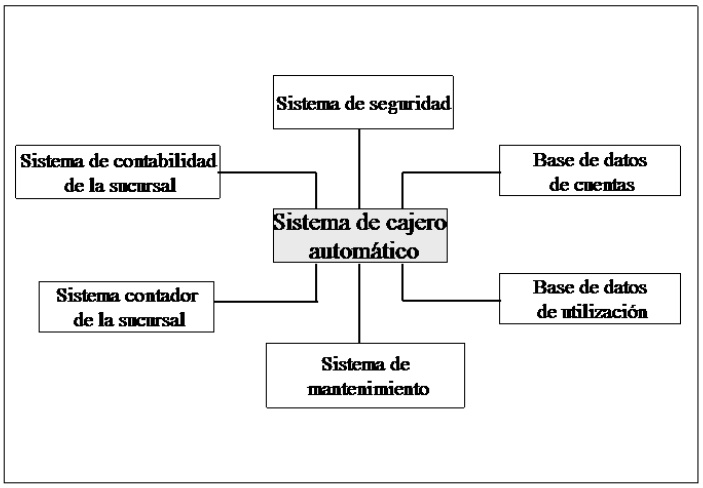


Figura 4.6 Modelo de contexto de un sistema de cajero automático.

2. **Perspectiva de Comportamiento**. Se utilizan para describir el comportamiento completo del sistema.

    a. *Modelos de Flujo de Datos.* Modelan el comportamiento de datos en el sistema, muestran como fluyen los datos a través de una secuencia de pasos de procesamiento. En la figura 4.7 se ilustra un modelo de flujo de datos para un sistema de pedidos.

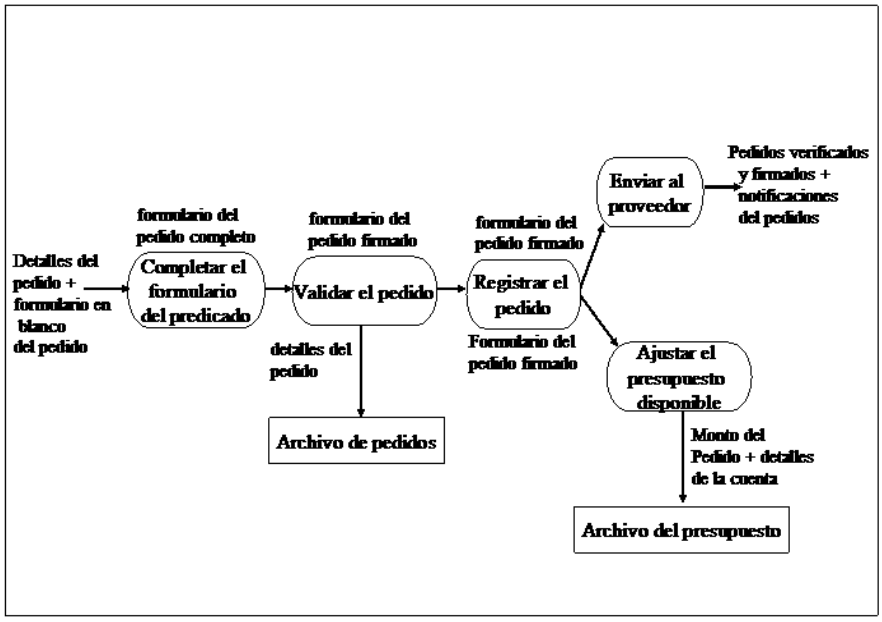


Figura 4.7 Diagrama de flujo de datos para el procesamiento de pedido.

b. Modelos de Máquinas de Estados. Modelan el comportamiento de un sistema en respuestas a eventos internos o externos, mostrando los estados del sistema y los eventos que provocan las transiciones de un estado a otro. Un ejemplo de este modelo se aprecia en la figura 4.8 para un sistema de un horno de microondas.

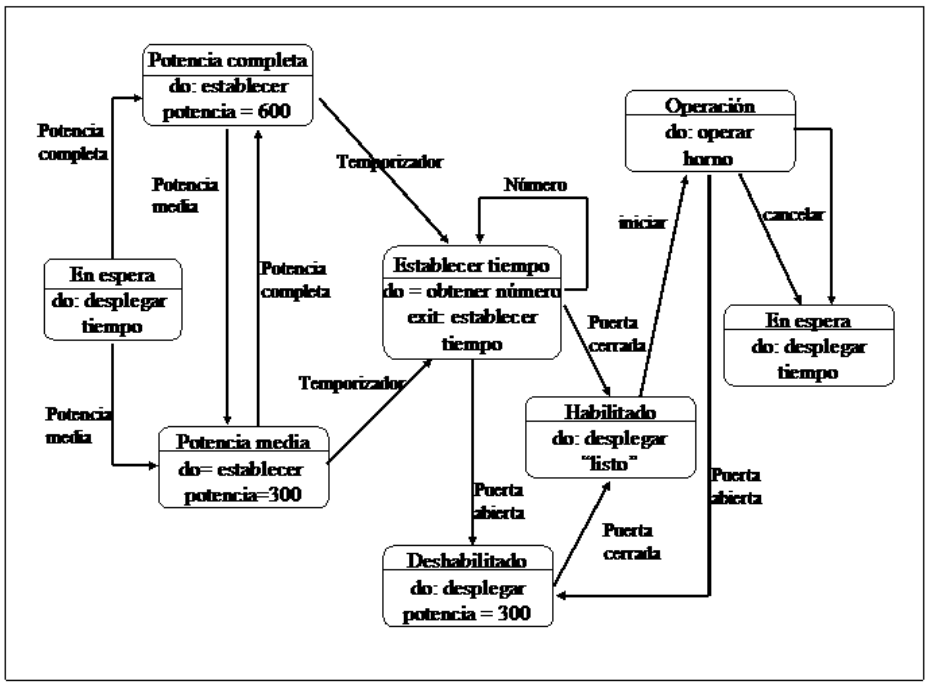


Figura 4.8 Modelo de máquina de estados de un horno de microondas sencillo.

3. **Perspectiva Estructural.** se modela la arquitectura del sistema o la estructura lógica de los datos procesados por éste.

   a. Modelo Entidad-Relación. Muestran las entidades de datos y sus atributos asociados y las relaciones entre estas entidades. La figura 4.9 muestra un ejemplo del modelo entidad-relación para un sistema de compra, donde se registra en la base de datos detalles del proveedor, la orden de compra y los artículos comprados.

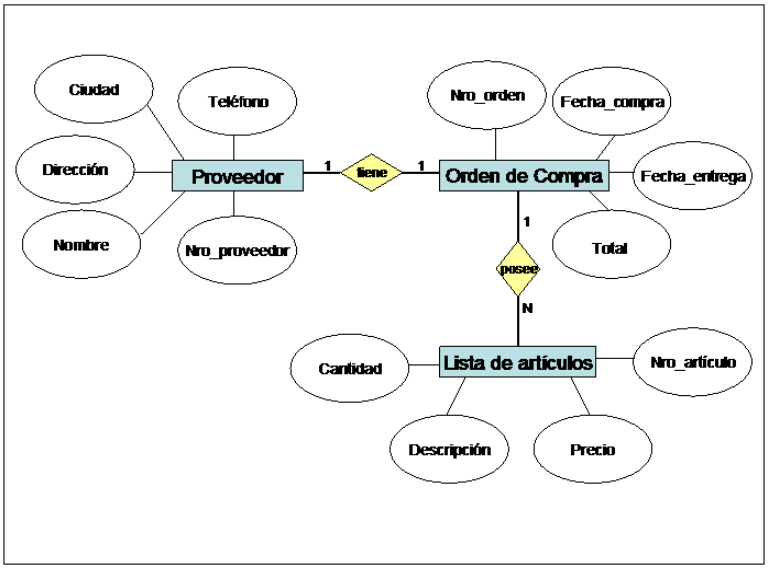


Figura 4.9 Modelo entidad-relación de un sistema de compra.

b. Modelo de Datos Semánticos. Muestra los modelos de datos en conjunto con los de flujo de datos para describir la estructura de la información que se esta procesando. En la figura 4.10 se aprecia un ejemplo del modelo de datos semánticos para un diseño de software.

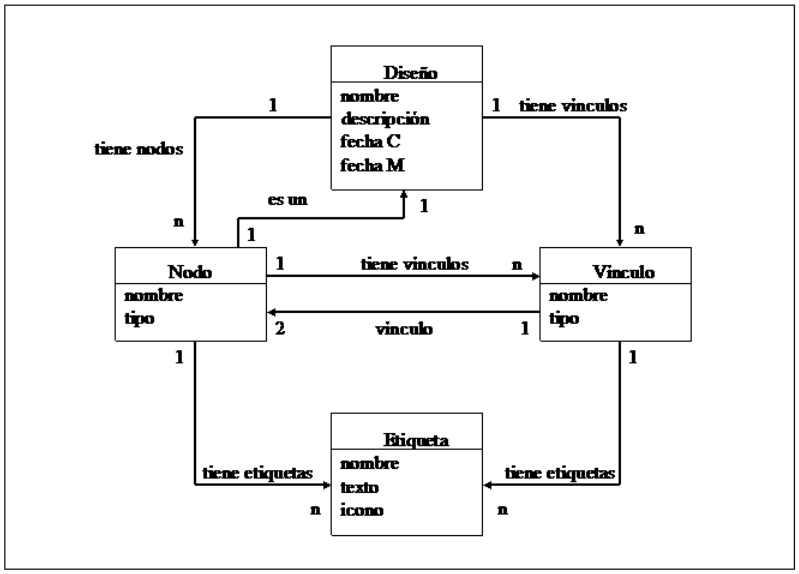


Figura 4.10 Modelo de datos semánticos para un diseño de software.

4. **Perspectiva Orientada a Objeto.** Combina los modelos de comportamiento y el estructural representando los datos mediante objetos y expresar los requerimientos del sistema utilizando casos de uso, realizan el diseño utilizando objetos y desarrollan el sistema en un lenguaje de programación orientado a objetos como lo es java o c++.

   a. *Modelos de Casos de Uso*. Permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requerimientos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso sirve como acuerdo entre el cliente y desarrolladores y proporciona las bases fundamentales para el análisis el diseño y las pruebas. Un modelo de casos de uso contiene actores, casos de uso y sus relaciones. En la figura 4.11 se muestra un ejemplo de un modelo de casos de uso para una contestadota automática en diagramas de UML.

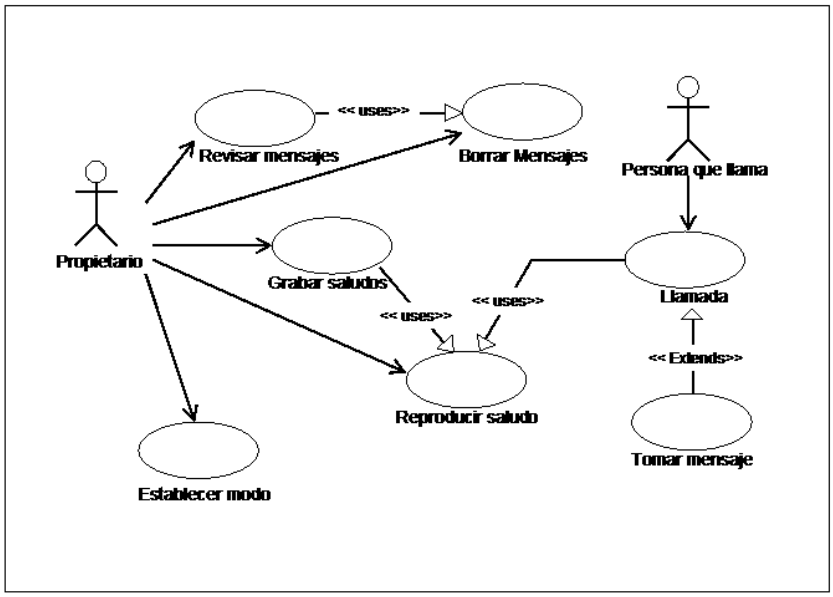


Figura 4.11 Modelo de Casos de uso para un sistema de contestadota telefónica automática.

b. *Modelos de Clases Jerárquicas (herencia)*. Identifican las clases de objetos que son importantes en el dominio del sistema. Organiza los objetos mediante una taxonomía. Una taxonomía es un esquema de clasificación que muestra la manera en que una clase de objetos esta relacionada con otras clases a través de servicios y atributos comunes. En la figura 4.12 se ilustra un ejemplo de este modelo, mostrando una parte de la jerarquía de clases para un sistema de biblioteca.

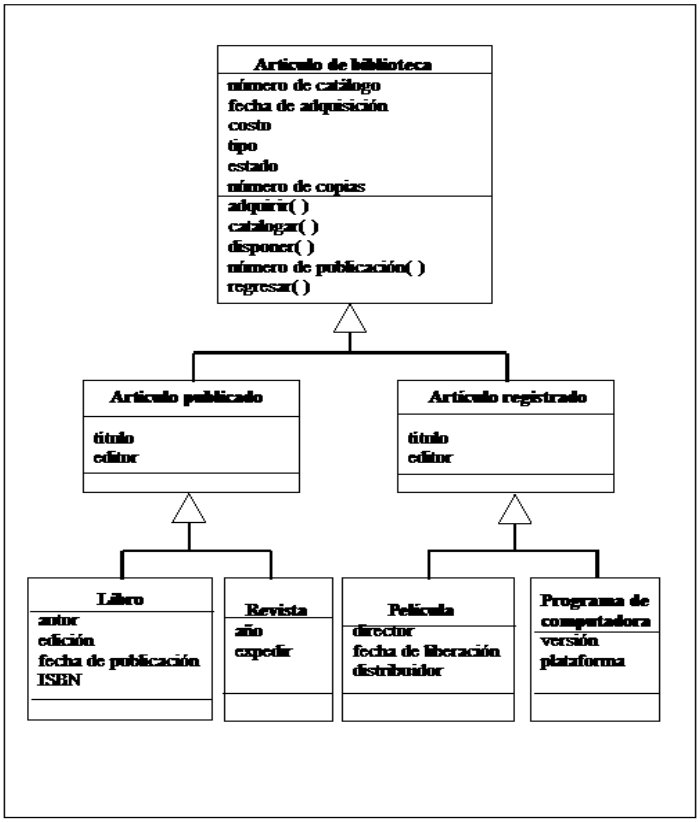


Figura 4.12 Jerarquía de clases para un sistema de biblioteca.

c. *Modelado de Comportamiento de Objetos (UML)*. En UML se utilizan diagramas de secuencia (figura 4.13) y diagramas de colaboración para mostrar el comportamiento e intercambio de mensajes de los objetos en el sistema.

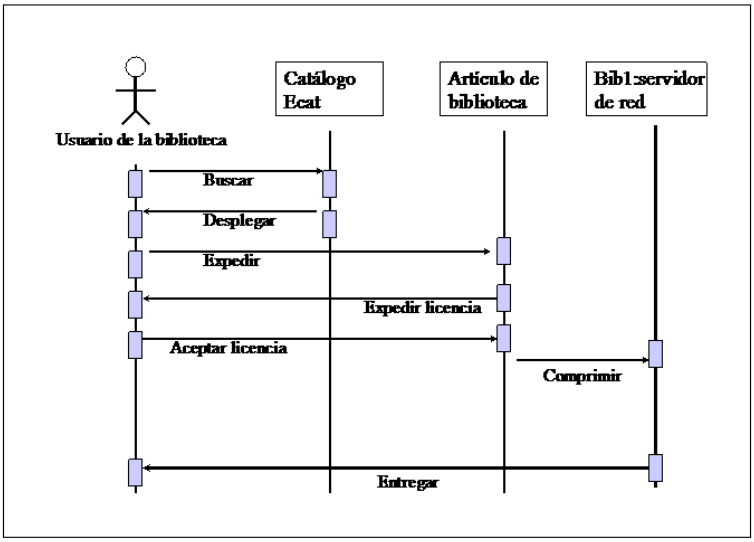


Figura 4.13 Diagrama de secuencia de un sistema de expedición de artículos electrónicos.

5. **CASE.** Es un conjunto de herramientas que ayudan a una fase particular del proceso de desarrollo de software como el diseño, la implementación o las pruebas. La figura 4.14 muestra el diagrama de una herramienta CASE.

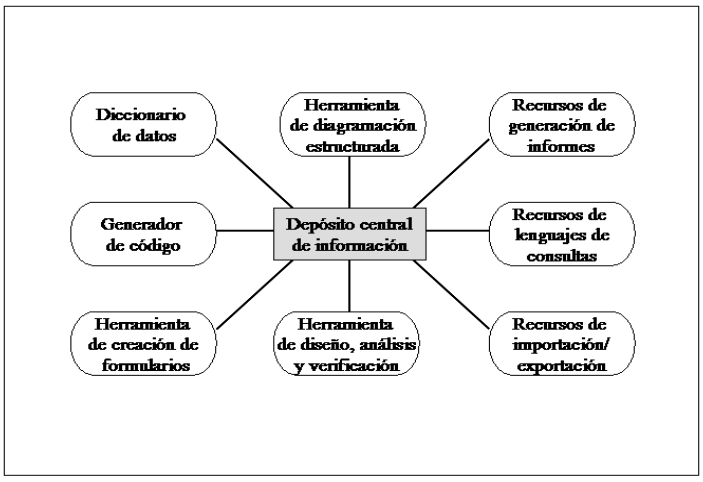


Figura 4.14 Banco de trabajo CASE para el análisis y diseño.

6. **Prototipo.** Un prototipo es un versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y de forma general enterarse más acerca del problema y sus posibles soluciones. La figura 4.15 muestra los dos tipos de prototipos existentes.

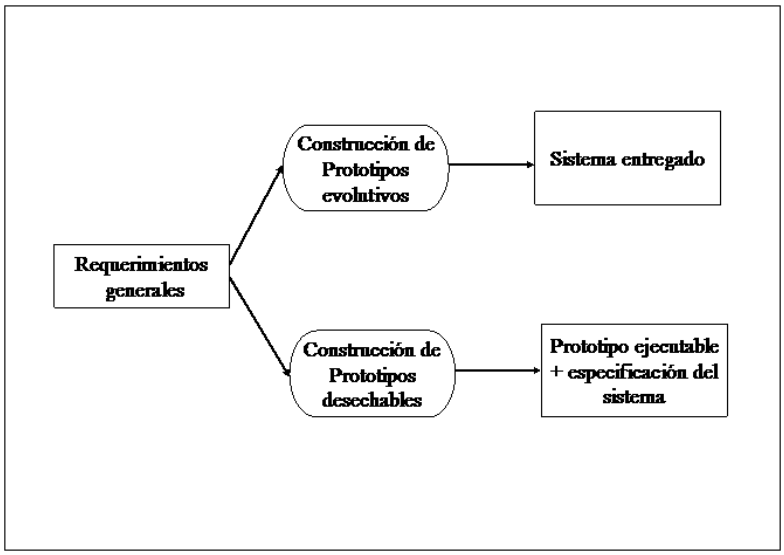


Figura 4.15 Construcción de prototipos evolutivos y desechables.

4.3.3 Especificación Matemática

La especificación de los requerimientos es redactada usando notaciones que se basan en conceptos matemáticos como el de máquinas de estados finitos o conjuntos. Estas especificaciones no ambiguas reducen los argumentos sobre la funcionalidad del sistema entre el cliente y el desarrollador. Sin embargo muchos clientes no comprenden las especificaciones formales y se rehúsan a aceptarlas como contrato del sistema.

4.4 Técnicas de Validación de los Requerimientos

La validación es la actividad que permite verificar sí los requerimientos obtenidos y analizados son los que definen el sistema que el cliente desea, y se lleva a cabo verificando cuidadosamente algunos criterios de validación que son establecidos, antes de que el documento de requerimientos sea usado como base en etapas posteriores del proceso de desarrollo del sistema. La validación y el análisis de requerimientos son procesos que tienen en común encontrar problemas con los requerimientos. La validación utiliza el documento de requerimientos completo mientras que el análisis trabaja con requerimientos incompletos. En esta actividad se pretende detectar y resolver conflictos entre requerimientos, verificar la consistencia e integridad entre los requerimientos, asegurar que los requerimientos puedan implementarse con la tecnología existente, considerando también el presupuesto y la calendarización del desarrollo del sistema y generar casos de prueba para los requerimientos.

4.4.1 Criterios de validación

Existen algunos criterios de validación que se deben llevar a cabo en el documento de requerimientos. Estos criterios incluyen:

1. *Verificaciones de consistencia.* Los requerimientos en el documento no deben contradecirse, es decir, no debe haber restricciones contradictorias o descripciones distintas de la misma función del sistema.
2. *Verificaciones de completitud*. En el documento de requerimientos deben estar los requerimientos que el usuario especifica, es decir deben contener todas las necesidades de los usuarios.
3. *Verificaciones de ambigüedad.* Los requerimientos deben verificarse, usando el conocimiento de la tecnología existente, para asegurar que puedan implementarse, considerando también el presupuesto y la calendarización del desarrollo del sistema.
4. *Verificaciones de rastreabilidad*. Los requerimientos del sistema siempre deben redactarse de tal forma que sea posible rastrearlos hacia su origen o hacia los requerimientos dependientes que lo detallan.

5. *Verificaciones de Correctibilidad*. Los requerimientos deben estar bien escritos, esto significa que puedan ser revisados por los clientes y desarrolladores.
6. *Verificaciones de Modificabilidad*. Los requerimientos deben ser especificados utilizando un lenguaje que permitan modificaciones.

Existen varias técnicas de validación de requerimientos que pueden utilizarse en forma individual o combinada.

4.4.2 Revisiones del Documento de Requerimientos

Este es un proceso manual, en el que se involucran varias personas para verificar el documento final de requerimientos, y participan tanto el personal del cliente como de los desarrolladores, en la revisión de anomalías y omisiones. El equipo de revisores debe verificar la consistencia de cada requerimiento y la integridad de estos como un todo, también se comprueban que el documento de requerimientos cumpla con los criterios establecidos de validación.

Los conflictos, contradicciones, errores y omisiones deben señalarse durante la revisión y registrarse formalmente.

4.4.3 Escenarios

Los escenarios son representaciones de como el usuario interactúa con el sistema. En este esquema se representan gráficamente las entradas, salidas, el flujo de datos y el comportamiento del sistema. Los escenarios pueden agregar detalles a un bosquejo de acción. En la figura 4.16 se representa un caso de uso para un usuario de una biblioteca que solicita un préstamo de un libro. Un escenario sería que el actor PrestatarioLibro:Juan Perez toma prestada la tercera copia del libro:UML Gota a Gota, de la biblioteca, cuando no tienen ningún otro préstamo. El sistema se actualiza de acuerdo a estas acciones.

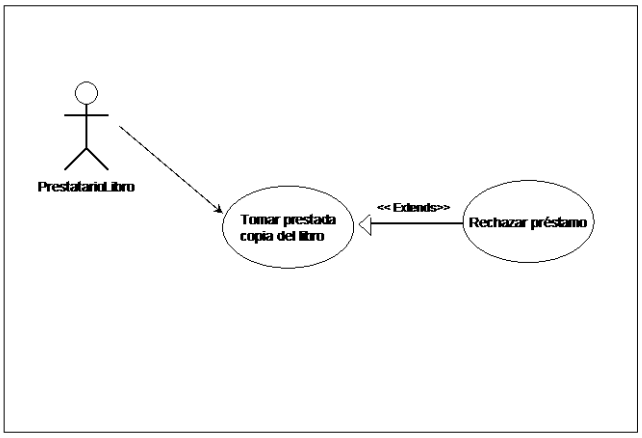


Figura 4.16 Escenario del caso de uso *tomar prestada copia del libro*.

4.4.4 Construcción de prototipos

En este enfoque de validación se presenta un modelo ejecutable del sistema a los usuarios y clientes, con el fin de que puedan realizar pruebas y comprobar si cumplen con sus necesidades reales. En la figura 4.17 se muestra un algoritmo del desarrollo de un prototipo evolutivo. Se basa en la idea de construir una implementación inicial, exponerla a los comentarios de los usuarios y refinarla mediante la repetición de las etapas hasta que se haya desarrollado un sistema adecuado.

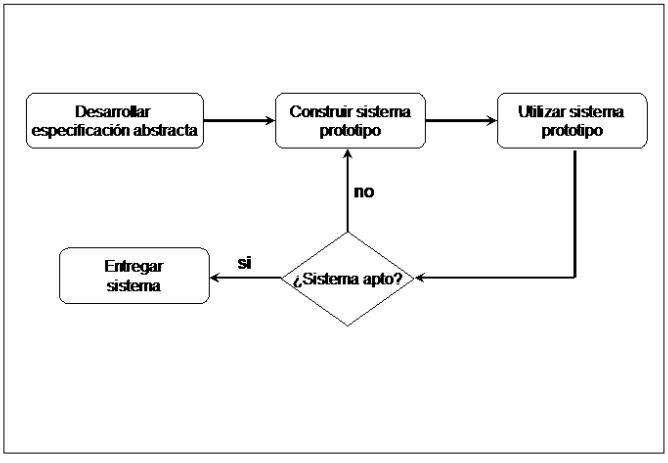


Figura 4.17 Construcción de prototipos.

4.4.5 Generación de casos de pruebas

Los requerimientos deberán ser probados. Si las pruebas pueden ser consideradas como parte del proceso de validación, esto frecuentemente permite describir los problemas en los requerimientos.

4.4.6 Análisis de consistencia automático

Si los requerimientos son expresados de manera automática mediante el uso de herramientas CASE, entonces dichas herramientas permitirán verificar la consistencia del modelo. Un analizador de requerimientos produce un informe de las inconsistencias descubiertas.

4.5 Técnicas de Administración del Documento de Requerimientos

Se conoce como Administración de los Requerimientos de software al proceso de comprender y controlar los cambios en los requerimientos del sistema. Este proceso se lleva a cabo junto con otros procesos de la ingeniería de requerimientos, y su planeación comienza al mismo tiempo que la obtención inicial de los requerimientos. La administración de los requerimientos debe comenzar tan pronto como esté lista la primera versión del documento de requerimientos.

4.5.1 Evolución de los Requerimientos

Existen diversas formas en que los requerimientos evolucionen o sean susceptibles a cambiar. Conforme se va desarrollando la definición de requerimientos se tiene mejor compresión de las necesidades del usuario, esto hace que el usuario se retroalimente de información y provoque que existan cambios en los requerimientos, más aun cuando se trata de sistemas demasiados grandes. El desarrollo de software puede llevar tiempo, y con esto el entorno del sistema y los objetivos del negocio puedan sufrir cambios, por lo tanto, los requerimientos deben adaptarse para reflejar estos cambios.

4.5.2 Administración del cambio en los Requerimientos

La administración del cambio de los requerimientos se aplica a todo los cambios propuestos en los requerimientos, en donde se evalúa el impacto y costo de dichos cambios. La figura 4.18 muestra los pasos a seguir en esta administración. La ventaja de utilizar esta administración, es que todos los cambios propuestos son tratados de forma consistente y que los cambios en el documento de requerimientos se realizan de forma controlada. Existen tres etapas de administración de cambio:

1. **Análisis del problema y especificación de cambios**. Se identifican los problemas en los requerimientos o la propuesta específica de cambios en los requerimientos. En esta etapa, la propuesta de cambio o el problema en los requerimientos, se analizan para verificar que si es válida, y entonces se realiza una propuesta de cambio de requerimientos más específica.

2. **Análisis del cambio y costos.** El efecto de un cambio propuesto es valorado. Para esto, se utiliza la información de rastreo y el conocimiento general de los requerimientos del sistema. El costo de realizar un cambio se estima en términos de las modificaciones hechas al documento de requerimientos y al diseño e implementación del sistema. Una vez completado el análisis, se toma la decisión de proceder o no con el cambio al requerimiento.

3. **Implementación de los cambio**. Se modifica el documento de requerimientos y, si es necesario también se modifica el diseño e implementación del sistema. El documento se organiza de tal forma que los cambios puedan llevarse a cabo sin que esto cause la redacción nuevamente del documento.

Si se requiere un cambio de forma urgente en los requerimientos del sistema, se podría modificar el sistema y después modificar el documento de requerimientos. Esto inevitablemente produciría un desfase entre la especificación de requerimientos y la implementación del sistema. Una vez realizados los cambios en el sistema, deben actualizarse los cambios necesarios en el documento de requerimientos, ya que si se deja pasar el tiempo se olvidan o se hacen de forma no consistente con los cambios del sistema.

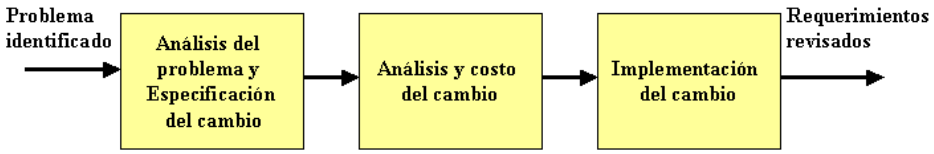


Figura 4.18 Administración de cambios en los requerimientos.

4.5.3 Políticas de rastreo

Las políticas de rastreo definen las relaciones entre requerimientos y las relaciones que existen entre los requerimientos con el diseño del sistema. Así mismo, estas políticas definen la forma en que se debe mantener su documentación. Cuando existen cambios en algunos requerimientos, se tiene que rastrear el efecto de estos cambios en otros requerimientos. El rastreo es una propiedad que facilita encontrar requerimientos relacionados. Según Sommerville [2] existen tres tipos de información de rastreo:

1. **La información de rastreo de la fuente**. Relaciona los requerimientos con los usuarios que propusieron los requerimientos. De esta forma cuando un cambio es propuesto, es posible consultar a estos usuarios.
2. **La información de rastreo de los requerimientos**. El documento de requerimientos es vinculado con otros requerimientos relacionados. Esta información es utilizada para evaluar de que forma el cambio de un requerimiento afecta a otros.
3. **Información de rastreo del diseño**. Vincula los requerimientos a los módulos de diseño. Esta información se utiliza para evaluar el impacto de los cambios hechos a los requerimientos en el diseño e implementación del sistema.

Con frecuencia se han utilizado matrices de rastreos para obtener estas tres tipos de información en el desarrollo de un sistema. En la matriz de rastreo, cada requerimiento está representado por una fila y una columna. Si existe una relación entre requerimientos esta se registra con una letra en una celda en donde se localiza la intersección entre los requerimientos. En la tabla 4.4 se indica el uso de letras para especificar el tipo de relación. La letra U, indica que el requerimiento en la fila utiliza los recursos del requerimiento indicado en la columna; una R significa que existe una relación débil entre los requerimientos. Las matrices de rastreos son útiles cuando se tiene que administrar un número pequeño de requerimientos, pero son muy difíciles de mantener cuando se trata de sistemas grandes con demasiados requerimientos.

Requerimientos	R1	R2	R3	R4	R5	R6	R7	R8
R1		U	R					
R2			U			R		U
R3	R			R				
R4			R		U			U
R5								U
R6		R		U				
R7								R
R8							R	

Tabla 4.4 Matriz de rastreo.

4.5.4 Soporte de herramientas CASE.

La administración de requerimientos comprende el procesamiento de grandes cantidades de información. Las herramientas que se utilizan van desde sistemas especializados de administración de requerimientos, hasta hojas de cálculo y sistemas de bases de datos sencillas.

4.6 Metodologías para el proceso de Ingeniería de Requerimientos

Algunas metodologías que se presentan a continuación tratan de abarcar todas las actividades del proceso de Ingeniería de Requerimientos, otras sólo comprenden ciertas actividades. En general, las metodologías nos indican una manera de cómo llevar a cabo sistemáticamente un proceso, para este caso, sería el proceso de la Ingeniería de Requerimientos.

4.6.1 Joint Application Design (JAD)

El Diseño Conjunto de Aplicaciones es una metodología cooperativa orientada al usuario, fue desarrollada por Chuck Morris y Tony Crawford en IBM en el año de 1977. La efectividad de esta metodología consiste en que el trabajo de obtención de requerimientos se realiza en una sesión de trabajo en la que participan todos los interesados, con el fin de presentar sus puntos de vistas, escuchar a los demás, negociar y ponerse de acuerdo en una solución mutuamente aceptable [8].

Entre los interesados que participan en una sesión JAD, se encuentran los siguientes:

1. Líder de sesión
2. Usuario representativo
3. Especialista en el contexto de la aplicación
4. Analistas
5. Representante de los sistemas de información
6. Cliente ejecutivo

El resultado de esta sesión es el documento JAD final, es un documento de especificación del sistema completo donde se incluyen definiciones de los datos, flujo de trabajo y pantallas de interfaz.

Existen cinco actividades necesarias para el desarrollo del JAD y son:

1. Definición del proyecto. El facilitador entrevista a los gerentes y usuarios con el fin de identificar el propósito, alcance y las metas del proyecto. En esta actividad se debe obtener un documento llamado ***Guía de Definición de Manejo*** (Management Definition Guide), que debe incluir: las funciones del sistema, la lista de suposiciones básicas y decisiones abiertas. Se identifican a las personas que tomarán decisiones finales sobre el sistema en las actividades posteriores (conocidas como el equipo JAD).

2. Investigación. En esta actividad se recopila información sobre el estado actual del proceso de negocios referente al sistema y se describen los flujos de trabajo. Finalmente se debe obtener una ***agenda de sesión***, y una especificación preliminar que lista las investigaciones realizadas, los nuevos procesos de negocio sugeridos, los datos involucrados, pantallas, reportes y lista de necesidades. Además, en esta actividad será necesario incluir algunos elementos de diseño para la clara representación de los elementos involucrados.

3. Preparación. En esta actividad se preparan todos los elementos necesarios para llevar a cabo "la sesión". Algunos de los elementos a preparar son: los elementos visuales de apoyo (acetatos, diapositivas, carteles) y el Documento de Trabajo (Working Document), documento base que contiene todos los documentos que se han acumulado durante la actividad de investigación y que se entregará a los miembros de "la sesión" como base para ésta.

4. La sesión. Esta actividad es el centro de JAD, donde el coordinador JAD guía al equipo para la creación de la especificación del sistema. El resultado de esta actividad debe ser una serie de formatos denominados Scribe (formularios), que servirán para construir el *Documento Final* y en donde se encuentran definidos el flujo de trabajo, los elementos de datos, las pantallas y los reportes del sistema.

5. Documento Final. Toda la información obtenida en la actividad anterior será utilizada para constituir el *Documento Final*. Se distribuirá a los miembros del equipo JAD en una reunión de revisión, el equipo JAD revisará y propondrá los ajustes necesarios. Finalmente, se harán los cambios necesarios y se recogerán las firmas de todos los integrantes del equipo JAD.

4.6.2 Cooperative Requirements Capture (CRC)

La técnica de captura cooperativa de requerimientos, esta metodología consiste en realizar una sesión de trabajo en la que participan todos los usuarios interesados, cada usuario debe presentar sus puntos de vistas, discutir con los demás para acordar una solución si es que existen diferencias entre los puntos de vistas de los demás participantes. En este aspecto es similar a JAD en su enfoque en sesiones de grupos de personas, pero con diferencias a JAD porque requiere la participación explícitamente de los interesados (stakeholders) quienes no pueden de otra manera estar involucrados en el desarrollo del sistema. Los participantes de las sesiones de CRC podrían incluir individuos con intereses financieros o de regulación en el producto final [14].

4.6.3 Objectory

La metodología Racional Objectory Process, es resultado de la integración de 3 metodologías orientadas a objetos (Rumbaugh 1991, Booch 1994 y Jacobson 1996) [5]. Abarca todo el proceso de desarrollo, aunque en este caso importa lo referente al análisis de requerimientos. El objetivo de la captura de requerimientos en Objectory es describir el ***qué*** debe hacer el sistema de software, definiendo el entorno y el comportamiento del sistema. La primera parte de la captura de requerimientos se realiza utilizando los Casos de Uso, para obtener un modelo en el que clientes y usuarios vean reflejados los servicios que el sistema va ha proveer. La notación utilizada en los casos de uso es sencilla y clara (vea casos de uso tema 4.1.5 apartado b, pag. 55).

Los modelos de diagramas de casos de uso dan una visión general del sistema, pero se puede agregar detalles utilizando escenarios que en Objectory se modelan aplicando diagramas de secuencia o de colaboración. Para cada caso de uso de este modelo es necesario elaborar varios escenarios, cada uno de ellos representará una situación particular de un caso de uso.

Es importante señalar que en Objectory todo el proceso de desarrollo de un nuevo software será manejado por los casos de uso, esto es, los requerimientos estarán presentes en todo momento, evitando con ello realizar productos de software incorrectos, es decir que no coincidan con las necesidades del usuario.

4.6.4 COHERENCE

Fue desarrollado en la Universidad de Lancaster, Gran Bretaña [20], su enfoque es humano–social, integra el análisis etnográfico para la integración social, y Objectory para el modelado de los requerimientos. Un esquema de los elementos que se consideran en Coherence se muestra en la Figura 4.19.

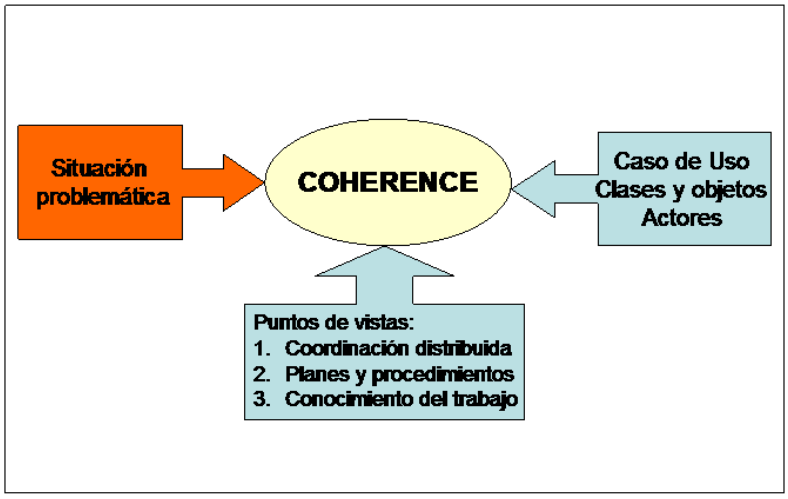


Figura 4.19 La metodología COHERENCE.

En Coherence se produce un enfoque sistemático que integra el análisis social con el análisis orientado a objetos para el diseño de sistemas de cómputo. El objetivo de Coherence es evitar que el usuario tenga que aprender nuevos elementos que ya han cubierto otras metodologías e integrar los elementos faltantes, como es el análisis social. En COHERENCE los puntos de vista representan el análisis social y son conectados con los elementos de Objectory (casos de uso, clases, objetos y actores).

1. **Punto de vista de coordinación distribuida**. La coordinación distribuida se usa para describir cómo las personas se coordinan para llevar a cabo sus tareas todos los días.

2. **Planes y procedimientos**. Se refiere a los diferentes objetos que son generados en un lugar de trabajo para documentar los procesos paso a paso. Se necesita aclarar cómo son utilizados los planes y procedimientos.

3. **Conocimiento del trabajo**. Se refiere a la forma en que se organizan las actividades para que sean inteligibles y lograr que las personas puedan hacer su trabajo.

4.6.5 SSM (Soft System Methodology)

La ***Metodología SSM*** fue desarrollada en Inglaterra por Peter Chekland en 1970, cuyo enfoque estudia el entorno en el que el sistema reside, investiga el porqué existe o podría existir el sistema, y como adaptarlo en un entorno general. Una vez que el sistema haya sido propuesto, las diferencias entre el sistema actual y el propuesto son examinadas. La metodología SMM se enfoca en un cambio revolucionario entre el sistema actual y el sistema futuro, determinando el estado final y como lograrlo, antes de estudiar las mejoras incrementales que podrían realizarse en el sistema actual. Aunque la notación es variada y flexible se hace hincapié en el uso de gráficos para lograr una rica imagen (rich picture) de situaciones [16]. Entre los gráficos propuestos están los que se muestran en la Figura 4.20 y representan lo siguiente:

Una **actividad determinada** es una acción tomada por un grupo de personas y que se basan en su experiencia del mundo real, en **intenciones** que se forman debido al medio ambiente en que se esté trabajando y las **restricciones** que se tengan. Las actividades determinadas repercutirán en un grupo de personas y podrán verse afectadas por otras actividades determinadas.

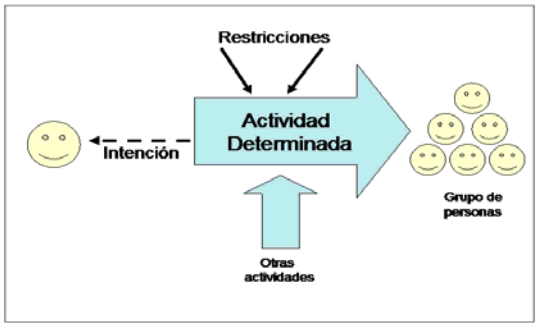


Figura 4.20 Gráficos utilizados por SSM.

La esencia de **SSM** es que reconoce que los sistemas están embebidos en un amplio contexto humano y organizacional, y fue uno de los primeros métodos en introducir el concepto de puntos de vista (llamados vistas del mundo), donde diferentes puntos de vista tienen diferentes percepciones del problema y la solución. En SSM existen siete etapas fundamentales, como se muestra en la Figura 4.21, y en seguida se explica brevemente cada etapa.

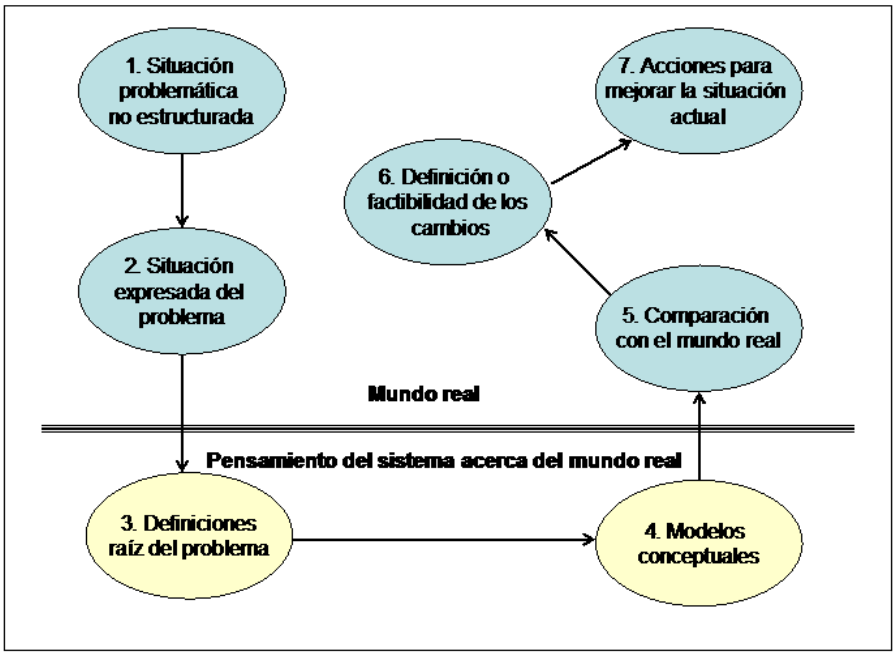


Figura 4.21 Los siete estados del modelo de SSM.

1. **Evaluación de la situación del problema**. El propósito es identificar quienes están involucrados, cuales son sus percepciones de la situación, cuáles son las estructuras organizacionales y qué procesos se llevan a cabo.

2. **Descripción de la situación**. Se describe la situación utilizando diagramas, lo que en SSM se le denomina una rica imagen (rich picture) en donde, por medio de iconos de varios tipos, se muestre el flujo de la situación actual, como se ilustra en la Figura 4.20.

3. Hacer **la selección de como ver paulatinamente la situación** de manera que se pueda producir **percepciones que den pié a definiciones raíz**. Lo esencial en este punto es producir

varias imágenes ricas en detalles de cada punto de vista diferente o subsistemas. Así, cada punto de vista relevante tendrá su definición raíz. Cada definición raíz en SSM es construida para posteriormente determinar su relevancia en el sistema a desarrollar. Para determinar si la definición raíz está bien escrita debe contener los elementos de la guía CATWOE, mnemónico correspondiente a las siglas que se explican en seguida:

C = Customer. Se trata de la persona que será beneficiaría o víctima del sistema.

A = Actors. Los actores, que llevarán a cabo las actividades definidas.

T = Transformation. Las transformaciones de entrada o salida.

W = Weltanschauung. Punto de vista del mundo real.

O = Owner. Propietario que tiene el poder de autorizar o rechazar el sistema.

E = Environmental. Restricciones del medio ambiente.

4. **Construir modelos conceptuales**. Un modelo debe corresponder a las actividades humanas de alguna definición raíz; de tal manera que hay una gran iteración entre las actividades 3 y 4.

5. **Comparar los modelos conceptuales con el mundo real**. Se hace una comparación de las actividades modeladas en el paso cuatro contra el mundo real.

6. **Identificar cambios factibles y deseables**. En esta etapa se deben investigar si los cambios son factibles culturalmente y sistemáticamente deseables. Esto se determina a través de reuniones conjuntas con la gente involucrada para recoger los diferentes puntos de vista y analizar la situación actual.

7. **Recomendaciones para la toma de acciones** que mejoren la percepción problema.

4.6.6 Análisis Estructurado

Las metodologías de análisis estructurado tomo relevancia en el trabajo realizado por DeMarco [2], en el que presentó y denominó los símbolos gráficos que ayudarían a los analistas crear modelos de flujo de información; sugirió heurísticas para utilizar esos símbolos, el uso de un **diccionario de datos** y **descripciones de procesamiento** como complemento a los modelos de flujo de información. Más tarde este método sufrió contribuciones interesantes como la Yourdon [3] y es conocido como **Análisis Estructurado Moderno**.

Esta metodología llama al análisis del comportamiento externo de un sistema como modelo de implementación del usuario, y se verá con mayor detalle, pero antes será necesario mencionar los modelos previos esencial y de ambiental y el Diccionario de Datos.

Antes de comenzar a explicar en qué consisten los modelos mencionados, se revisará la notación de esta metodología; es variada, pero en las etapas que se verán, sólo se usan los Diagramas de flujo de Datos (DFD).

Los DFD son muy sencillos: rectángulos para terminadores (personas, reportes), rectángulos redondeados y flechas que conectan los otros iconos e indican el flujo de los datos que van escritos junto a ellas. Existen conexiones inválidas de iconos y son: dos terminadores o dos almacenes. Un ejemplo de DFD es el de la Figura 4.22.

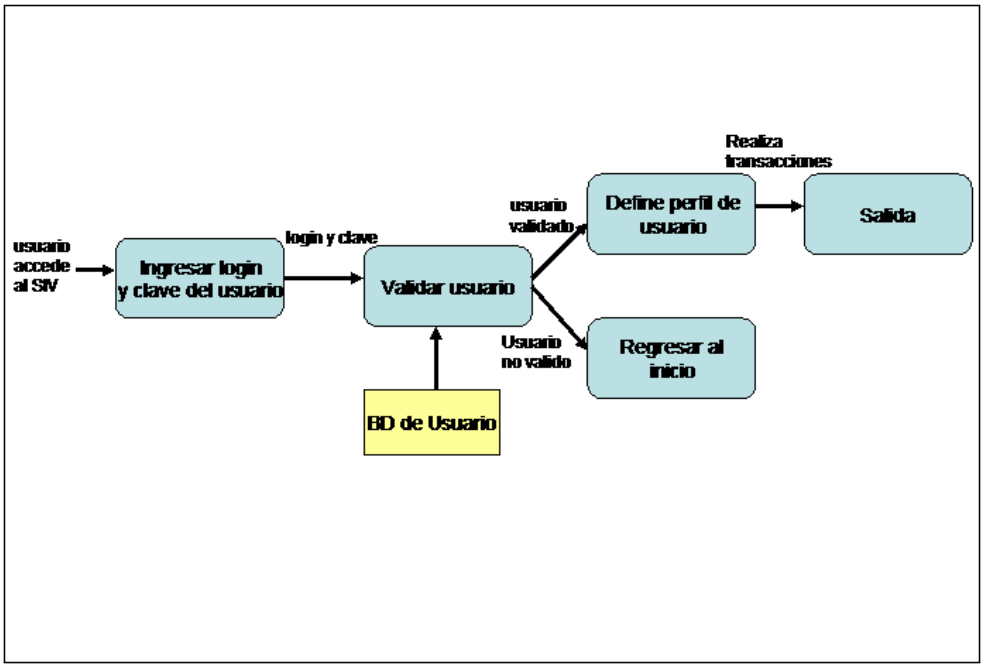


Figura 4.22 Ejemplo de Diagrama de Flujo de Datos.

Modelo esencial. Es un modelo de lo qué el sistema debe hacer para satisfacer los requerimientos del usuario. Para lograrlo se debe suponer un costo nulo de la tecnología y no tratar de escribir la especificación de los procesos. Para construir el modelo esencial deben considerarse los siguientes pasos:

1. Realizar varios DFD de temas separados y de tamaños medianos.

2. Los datos que fluyen de un proceso a otro deben empaquetarse (varios datos juntos) según las necesidades de cada proceso.

3. No detallar los procesos en subprocesos, sólo deben estar los esenciales.

4. Eliminar procesos cuyo objetivo sea únicamente transportar datos de un lugar a otro.

5. No incluir los procesos de verificación de datos de entrada o salida.

6. Englobar los almacenamientos que intervengan en el mismo proceso.

7. Eliminar datos que no intervengan en ningún proceso y aquellos datos y almacenes que puedan ser derivados.

8. Evitar poner aquellos almacenes que son de apoyo para la implantación.

Modelo ambiental.

En inglés es *Environment Model*, en este se debe definir los elementos que son parte del sistema y los que no lo son; para ello hay que tener en mente que no importa cuán importante sea este modelo pues, al fin y al cabo es sólo una parte de los procesos de la empresa y como tal hay que fijar sus límites o fronteras.

En el modelo ambiental deben definirse las interfaces entre el sistema y el resto de los elementos que lo rodean; necesita contar con los siguientes tres elementos:

1. Enunciado del propósito del sistema.

2. Diagramas de contexto. Es un DFD que incluye personas, datos, sistemas que entran o salen e interactúan con el sistema a realizar.

3. Lista de eventos. Se debe hacer la narrativa de los estímulos que ocurren fuera del sistema y a los cuáles éste debe responder.

En un diagrama de contexto, las partes que corresponden al sistema deben distinguirse de alguna manera de las que son externas, para ello se propone que se enmarquen las partes que corresponden al sistema.

Diccionario de Datos.

Aunque un Diccionario de Datos (DD) no aparece únicamente en esta metodología, sí se le da bastante importancia y se va indicando a cada momento cómo debe ser llenado y con qué. Así por ejemplo, cuando se indica que en el modelo esencial deben empaquetarse lo datos, señala que cada paquete debe ser detallado en el DD. Además propone la notación BNF para su definición.

Modelo de Implementación del usuario.

En este modelo deben definirse las interfaces del sistema con el medio ambiente que lo rodea y se realizará con ayuda de un diseñador. El Modelo de Implementación del usuario consiste en:

1. *Determinar las metas de la automatización*. Al llegar a este punto ya se han definido las actividades esenciales (funciones) y también todos los datos esenciales Lo que ahora se definirá son funciones y datos que se realizarán automáticamente así como aquellas que se manejarán manualmente. En la elección pueden intervenir varios factores, entre otros están los temores del usuario, las condiciones ambientales y económicas. Los DFD serán modificados para señalar cuáles procesos son manuales y cuáles no; para distinguirlos se pueden utilizar sombreados o colores.

2. *Determinar la Interfaz del usuario*. Ésta es la que lleva más tiempo, debido a que consta de cuatro temas:

    2.1. Escoger dispositivos de entrada y salida.

    2.2. El formato de todas las entradas, incluyendo diagramas de transición para el manejo de la entrada.

    2.3. El formato de todas las salidas, incluyendo diagramas de transición para obtener la salida.

    2.4. La secuencia y los tiempos de todas las entradas y salidas de un sistema en línea.

3. *Identificar las actividades manuales para el soporte del sistema*. Dado que en el modelo esencial se supuso una tecnología perfecta y de precio nulo, aquí se deben determinar las fallas que pueden suceder y la forma en que se deben solucionar:

3.1. Fallas. Puede haber en la entrada de los datos (perdidos o duplicados), errores lógicos, de conexión entre partes del sistema, daño de dispositivos de almacenamiento o de manejo de entradas y salidas.

3.2. Soluciones. Redundancia de dispositivos y manejo de transacciones.

4. *Especificar las restricciones operacionales*. El equipo de desarrollo tendrá que decidir el hardware, sistema operativo, facilidades de comunicación, lenguaje de programación y las estrategias de diseño. Para ello se deben definir las restricciones operacionales, las cuales el modelo esencial no tomó en cuenta:

4.1. Volumen de datos. Qué tantos datos se manejan y cuánto esperan que crecerá dicho volumen.

4.2. Tiempo de respuesta. Debe ser puesto en términos absolutos ayudados de porcentajes.

Ejemplo: en el 90% de los casos el proceso debe responder en 2 segundos como máximo.

4.3. Restricciones políticas existentes que se sobreponen a las decisiones de implantación.

4.4. Medio ambiente físico. Temperatura, humedad, interfaz eléctrica, peso y tamaño del equipo, entre otras.

4.5. Restricciones de seguridad y confiabilidad. Tiempo medio entre fallas y tiempo medio entre reparaciones.

4.6. Restricciones de acceso.

4.6.7 Metodología de DSED

La metodología de DSED (Desarrollo de Sistemas Estructurados de Datos) o de Warnier-Orr fue comenzada por Warnier en 1974 y continuada por Orr en 1977, su enfoque es sobre los datos y lo que busca es tener una herramienta para representar la jerarquía de la información y el flujo que mantiene la información en el dominio del problema [3].

La notación empleada es sencilla y clara, consiste en llaves, círculos y flechas. Las primeras se utilizan para representar la jerarquía de los datos en los Diagramas de Warnier. Un ejemplo de su uso se ve en la Figura 4.23, donde se describe el contenido de una factura que tiene capacidad de incluir hasta cinco diferentes objetos comprados.

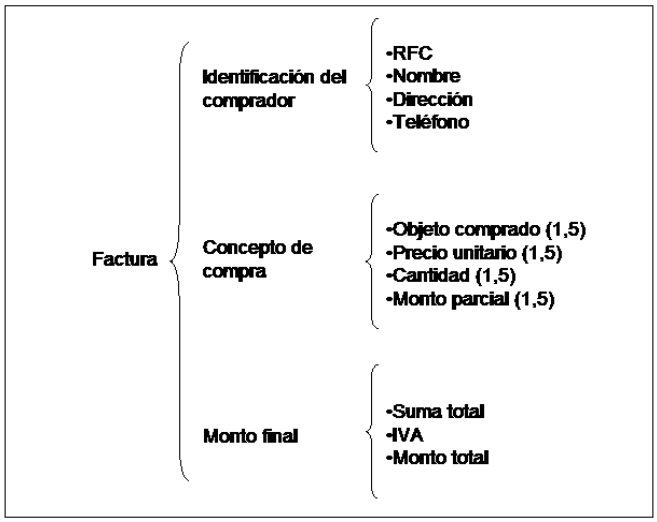


Figura 4.23 Diagrama de Warnier.

Los círculos representan productores o consumidores (persona, máquina, otro sistema), las flechas se utilizan para conectar los círculos, esto es los productores y consumidores. Como ejemplo se tiene el de la Figura 4.24.

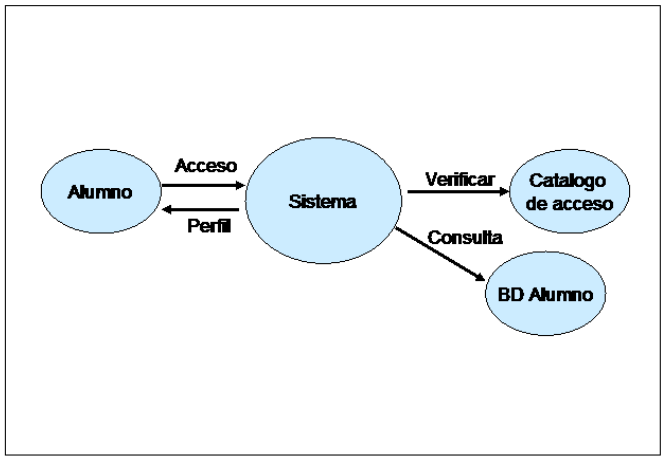


Figura 4.24 Diagrama de entidades para acceso al usuario Alumno.

La metodología comprende los siguientes pasos:

1. **El análisis del contexto de la aplicación**. Se busca definir los elementos de información, los productores y consumidores de esa información y los puntos de vista sobre la información de cada productor o consumidor.

2. **La definición de las funciones de aplicación**. Se definen las funciones que deben implementarse para el sistema.

3. **Definición de los resultados de la aplicación**. Se incluye la construcción de un prototipo en papel de las salidas deseadas del sistema y éstas deben corresponder a los diagramas de Warnier definidos en el paso uno.

4. **Requerimientos físicos**. Se hace una lista de los requerimientos del comportamiento del sistema divididos en requerimientos de: comportamiento, fiabilidad, seguridad, hardware e interfaces de comunicación.

4.6.8 COLOR-X

Es una herramienta automática y a su vez una metodología basado en la lingüística computacional y el análisis orientado a objetos. Comprende varias etapas de la Ingeniería de Requerimientos: especificación, validación y verificación de requerimientos. COLOR-X es un acrónimo de **C**onceptual **L**inguistically based **O**bject oriented **R**epresentation language for information and communication systems, más una X que agrega el autor J.F.M. Burg. La metodología fue realizada como una tesis doctoral en la Universidad de Vrije, Holanda [17, 21].

La forma en que COLOR-X trabaja se muestra en la Figura 4.25. Todos los procesos son automáticos o semiautomáticos, es decir, son en línea con el usuario. Se parte del hecho de que ya existe al menos una primera redacción de los requerimientos del sistema.

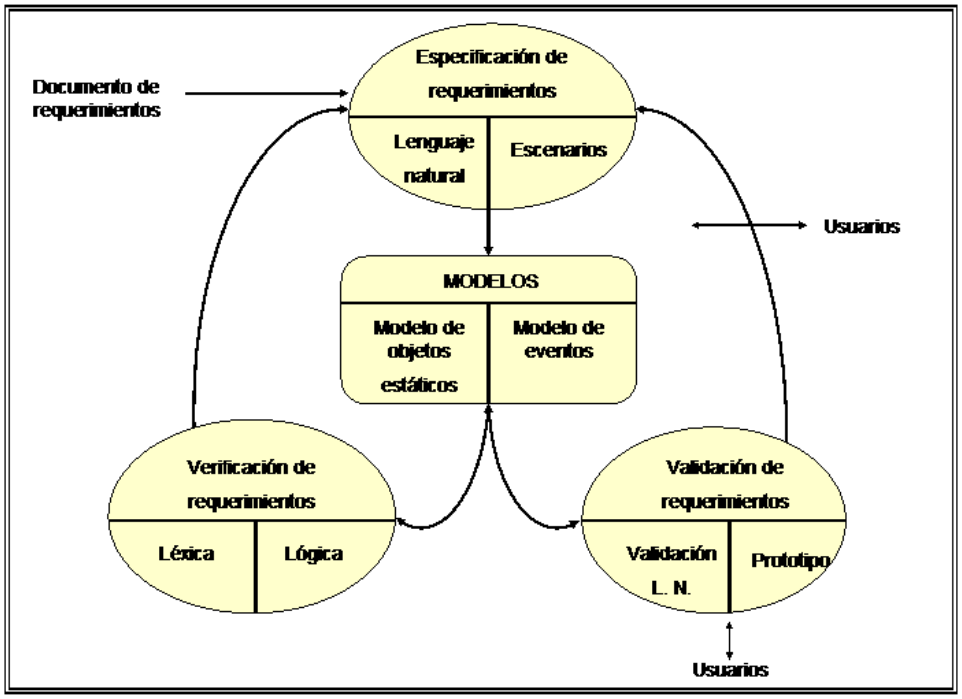


Figura 4.25 Estructura COLOR-X.

Cada fase está representada en la Figura 4.25 por un círculo, que a su vez se divide en otras etapas y que utilizan los modelos conceptuales de COLOR-X, los cuales son los escenarios y modelos de lenguaje natural. Todas las fases son ejecutadas iterativamente para actualizar la ERS de acuerdo con los resultados de la validación y la verificación. Los usuarios pueden dar información adicional en cualquier momento.

4.6.9 RARE-IDIOM

Esta metodología presenta el enfoque de reuso, el cual ha tomado fuerza en la última década. Se basa en las propiedades estructurales de los documentos de requerimientos y los procesos realizados para su producción. La metodología propone varias formas de reusar los requerimientos y sus artefactos asociados, todo descrito en términos de patrones de reuso de requerimientos. RARE-IDIOM fue elaborada por Jacob Cybulski en la Universidad de Melbourne en Australia [22], y es un acrónimo de **R**euse-**A**ssisted **R**equirement **E**ngineering with **I**nformal **D**ocument **I**nterpreter **O**rganizer and **M**anager, su idea es reutilizar en todo momento a partir de que el sistema es solicitado por el cliente partiendo de ERS anteriores como se muestra en la Figura 4.26.

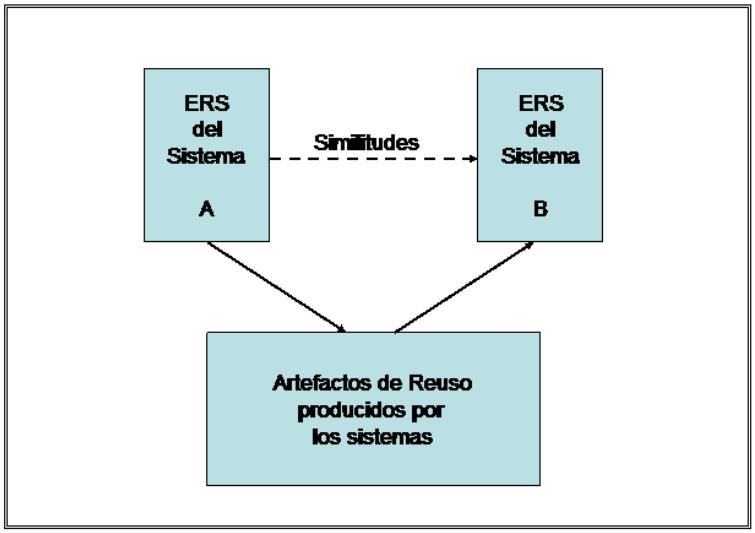


Figura 4.26 Reuso de ERS anteriores.

Además propone un nuevo ciclo de vida de Ingeniería de Requerimientos que introduce el reuso y que quedaría como se muestra en la Tabla 4.5. En la fase tres de la Tabla 4.5 se propone la aplicación de varios tipos de técnicas: analíticas, de procesos y por productos.

Para la Generación y Evaluación de Alternativas RARE-IDIOM propone los siguientes pasos:

1. Clasificar los requerimientos y artefactos reutilizables del sistema actual en términos precisos para que tal clasificación pueda ser usada al compararlos con los artefactos reutilizables anteriores.

2. Precisar la búsqueda de artefactos candidatos mediante la selección de categorías de requerimientos en términos del dominio del problema, tipos de requerimientos y su semántica.

3. Reducir el alcance de la búsqueda de artefactos al seleccionar categorías de artefactos en términos de la solución del problema.

4. Comparar el texto de la descripción del problema contra la colección estructurada de artefactos reusables, aspecto por aspecto.

5. Comparar los beneficios y daños de seleccionar cada artefacto candidato para refinar un enunciado de requerimiento actual.

6. Seleccionar la variante que mejor empate con el requerimiento actual.

7. Continuar seleccionando categorías de requerimientos alternativas mediante la generalización de los requerimientos actuales hasta encontrar un artefacto adecuado o ya no haya posibilidad de hallarlo.

Etapa	Actividad	Elemento de reuso
Elaboración de necesidades y objetivos.	Modelado de la empresa	Modelo de la empresa
Obtención de requerimientos.	Obtención de habilidades y recursos. Reuso por medio de los documentos de requerimientos	Fuentes de información. Plantillas de documentos y estándares. Información fuente en bruto. Conceptos del dominio. Frases textuales.
Especificación de requerimientos y modelado.	Refinamiento de artefactos. Adaptación e integración de procesos. Análisis y adaptación de patrones.	Comentarios y anotaciones. Modelos de requerimientos. Patrones.
Generación y evaluación de alternativas.	Restricción del dominio del problema. Restricción del dominio de soluciones. Evaluación y selección.	Especificación de sistemas de software anteriores.
Verificación y validación de requerimientos	Aplicación de reglas de verificación. Validación.	Reglas de verificación

Tabla 4.5 Reuso mediante las actividades del proceso de obtención de requerimientos.

4.6.10 El método SADT

El método SADT (Structure Análysis and Design Technique, **Técnicas de Análisis y Diseño Estructurado**), fue desarrollado en los años 70's [3]. Este método está basado en el modelo de flujo de datos y muestra los sistemas como un conjunto de actividades interactuando. SADT toma sus siglas de la diagramación con rectángulos y flechas del análisis estructurado (SA) y de las técnicas de diseño (DT). SADT ha sido muy usado en la especificación de una gran variedad de productos, especialmente en proyectos de gran escala.

La notación consiste de un rectángulo que representa algunas actividades del sistema y cuatro grupos de flechas, cada grupo con diferente significado semántico, como se visualiza en la figura 4.27.

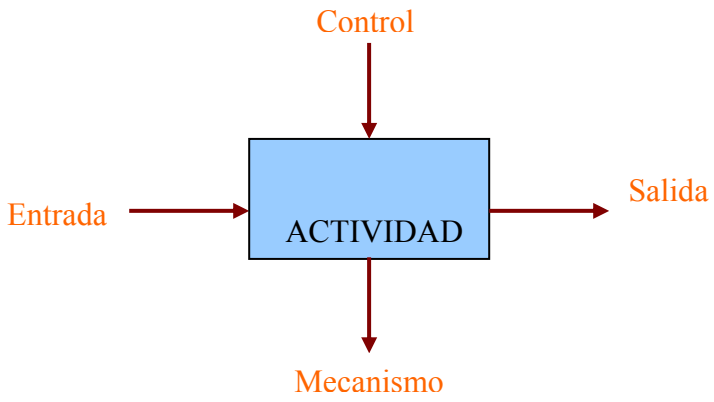


Figura 4.27 Notación SADT.

La flecha de la derecha representa la salida de datos, la flecha de arriba el control de la información o datos para facilitar el proceso. SADT integra el análisis y diseño de los requerimientos y los incorpora en su notación. La flecha de abajo representa el mecanismo o algoritmo por el cual el proceso es establecido. SADT descompone el problema en un conjunto de jerarquías de diagramas, cada uno compuesto de un conjunto de rectángulos y flechas. Cada nivel inferior es documentado separadamente y representa el refinamiento del nivel anterior. El nivel más abstracto es llamado el diagrama de contexto. Un rectángulo, representa las actividades del sistema, junto con un conjunto de entradas y salidas que constituyen el punto inicial para la descomposición funcional del sistema.

SADT no tiene una definición explicita de puntos de vista, en cambio los puntos de vista son una extensión de esta técnica de modelado. Los puntos de vista SADT representan las fuentes y repositorio de los datos. SADT es más bien un método intuitivo más que un punto de vista, es decir, que los puntos de vista sólo aparecen en el nivel de contexto y en ninguna otra etapa del método. El análisis de puntos de vista en SADT sólo los considera como fuentes y repositorio de datos.

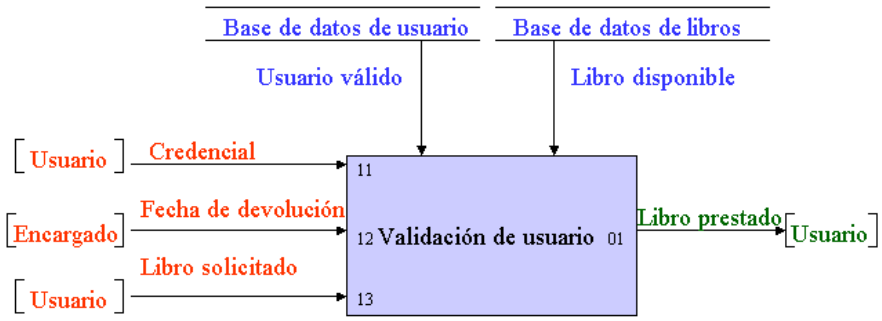


Figura 4.28 Diagrama de actividades para la Biblioteca.

4.6.11 El método CORE

El CORE (Controlled Requeriments Expression, **Expresión de Requerimientos Controlados**), fue desarrollada para la fuerza aérea Británica en los años 70´s y por el diseñador de sistema Mullery [2]. De la misma forma que SADT, CORE está basado en la descomposición funcional, sin embargo, a diferencia de SADT, CORE está basado en puntos de vista. CORE ha sido usado extensamente por la industria aeronáutica Europea. CORE es uno de los pocos métodos de requerimientos que adopta explícitamente un enfoque de puntos de vista para formular los requerimientos. CORE define sus puntos de vista en dos niveles: el primer nivel comprende todas las entidades que interactúan o son afectadas de alguna manera por el sistema. CORE proporciona las directrices para identificar los puntos de vista funcionales y no funcionales. El segundo nivel distingue entre los puntos de vista definidos internamente de los puntos de vista externos.

Los puntos de vista definidos son subprocesos del sistema, los cuales son, de forma top-down (de arriba hacia abajo), y los puntos de vista externos son entidades que interactúan indirectamente con el sistema propuesto. El método CORE consiste en siete pasos interactivos:

1. Identificación de los puntos de vista
2. Estructuración de los puntos de vista
3. Recolección tabular
4. Estructuración de los datos
5. Modelado simple de los puntos de vista
6. Modelado combinado de los puntos de vista
7. Análisis de restricciones

4.6.12 El método VOSE

El método VOSE (Viewpoint-Oriented System Engineering, **Ingeniería de Sistemas Orientada a Puntos de Vista**), es una estructura o enfoque para integrar métodos de desarrollo. Este enfoque maneja el ciclo de desarrollo del sistema completo [2]. VOSE fue desarrollado en el Colegio Imperial de Londres en los años 1990's. La filosofía de este enfoque consiste en que, el desarrollo de software involucra la participación de muchos expertos en el proceso de desarrollo de software y en el área de la aplicación. Cada participante puede tener responsabilidades, como establecer cambios en los objetivos del negocio y producir estos cambios en el desarrollo y evolución del software. Para manejar y capturar estas múltiples perspectivas, este enfoque se auxilia del uso de puntos de vista, para dividir y distribuir las actividades y el conocimiento de los participantes. Los puntos de vista capturan el rol y la responsabilidad de un participante en una etapa específica del proceso de desarrollo de software.

Los puntos de vista son identificados por el rol del participante, el área relacionada a su interés y el conocimiento acerca del dominio de la aplicación. Este conocimiento es encapsulado en un punto de vista y representado utilizando un esquema simple, llamado "estilo".

Un punto de vista en el método VOSE, puede representarse mediante una plantilla que describe un estilo o un esquema representativo. El punto de vista expresa lo que es visto, el dominio, una especificación, un plan de trabajo que define las condiciones bajo las cuales la especificación puede ser cambiada y una bitácora de trabajo. La figura 4.29 muestra los espacios en una plantilla de un punto de vista estándar en el método VOSE.

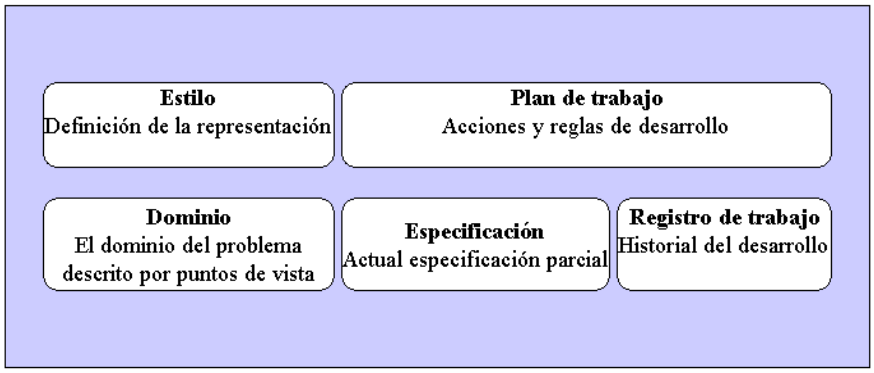


Figura 4.29 Espacios de un punto de vista, plantilla en VOSE.

Los puntos de vista VOSE pueden estar organizados en configuraciones, las cuales son colecciones de puntos de vista relacionados. La configuración para un problema de dominio hipotético, puede consistir de plantillas con diferentes estilos "viendo" la misma división del dominio del problema, o plantillas con el mismo estilo "viendo" diferentes divisiones del dominio del problema. El sistema final es una combinación de configuraciones con todos los conflictos resueltos.

En el caso de plantillas con el mismo estilo "viendo" diferentes divisiones, la resolución del conflicto involucrado asegura que hay consistencia en el flujo de información entre las diferentes divisiones. Sin embargo, donde la configuración consiste de diferentes plantillas con diferentes estilos, "viendo" el mismo dominio, los estilos pueden ser escogidos cuidadosamente para asegurar que tengan un alto grado de correspondencia.

4.6.13 El método VORD

El método VORD (Viewpoint-Oriented Requirements Definition, **Definición de Requerimientos Orientado a Puntos de Vista**), es sobre todo propuesto para especificar sistemas interactivos, pero también puede ser usado para especificar otras clases de sistemas [11]. El método VORD está basado en puntos de vista que se enfocan en usuarios involucrados en aspectos organizacionales. El modelo orientado a puntos de vista es orientado a servicios. El sistema da servicios a los puntos de vista y los puntos de vista pasan información de control y parámetros asociados al sistema. Los puntos de vista proyectan las clases de usuarios finales de un sistema o de otro interconectado. Los puntos de vista que estructuran el núcleo del modelo, son conocidos como puntos de vista directos. Por el hecho de que no todos los requerimientos son derivados de gente o subsistemas que interactúan con sistemas especificados. Los puntos de vista que respecta a sistemas externos que tienen influencia en el dominio de la aplicación también son considerados y son llamados puntos de vista indirectos.

1. **Puntos de vista Directos**. Estos corresponden directamente a clientes que reciben servicios del sistema y envían información de datos y de control al sistema. Pueden ser cualquier sistema usuario/operador u otro subsistema que interactúa con el sistema analizado.
2. **Puntos de vista Indirectos**. Los puntos de vista indirectos tienen un interés en algunos o todos los servicios del sistema, pero no interactúan directamente con él, estos pueden generar requerimientos que restringen los servicios entregados a puntos de vista directos.

Los puntos de vista indirectos tienen una importancia en los puntos de vista de la ingeniería (estos se refieren al diseño e implementación del sistema), mediante los puntos de vista organizacional (se refieren a la influencia de organización), a puntos de vista externos (se refieren a la influencia del sistema en el ambiente externo). El método VORD está basado en tres principales pasos iterativos llamados:

1. Identificación y estructuración de los puntos de vista.
2. Documentación de los puntos de vista.
3. Análisis y especificación de los requerimientos de puntos de vista.

En el primer punto se trata de identificar los puntos de vista, y son declaraciones abstractas de las necesidades de la organización. El segundo paso consiste en documentar los puntos de vista identificados en el paso uno. La documentación consiste en:

1. Etiquetar y describir el punto de vista identificado.
2. Rastrear los tipos de puntos de vista como directos o indirectos.
3. Atributos que caracterizan a los puntos de vista en el dominio de la aplicación.
4. Requerimientos de puntos de vista, estos incluyen un conjunto de servicios requeridos, requerimientos de control y requerimientos funcionales.
5. Escenarios de eventos que describen la interacción entre los puntos de vista y el sistema propuesto.
6. Historial de puntos de vista que describen la evolución de los requerimientos de los puntos de vista.

El tercer paso se refiere con la identificación de errores y conflictos y su solución. El resultado final es el documento de la especificación de requerimientos.

En la figura 4.30 se muestra el proceso del modelo iterativo VORD. Los procesos son mostrados en rectángulos con esquinas redondeadas y los productos en rectángulos con esquinas cuadradas. Cada producto puede ser visto como el control para un proceso de revisión.

La notación grafica usada para representar los puntos de vista en el VORD se muestra en la figura 4.31 Los puntos de vista se representan en rectángulos, se identifican como se muestra en la esquina superior izquierda y se etiquetan a partir de la mitad de abajo del rectángulo, el tipo de punto de vista o rastreo se muestra en la esquina superior derecha y su especialización son rectángulos contiguos por el lado derecho, señalado por una flecha que parte de izquierda a derecha. Cada punto de vista tiene uno o más tipos especializados. Un punto de vista especializado comparte ciertos servicios y atributos con los puntos de vista padre, pero puede recibir servicios adicionales o imponer sus propias limitaciones sobre los servicios heredados.

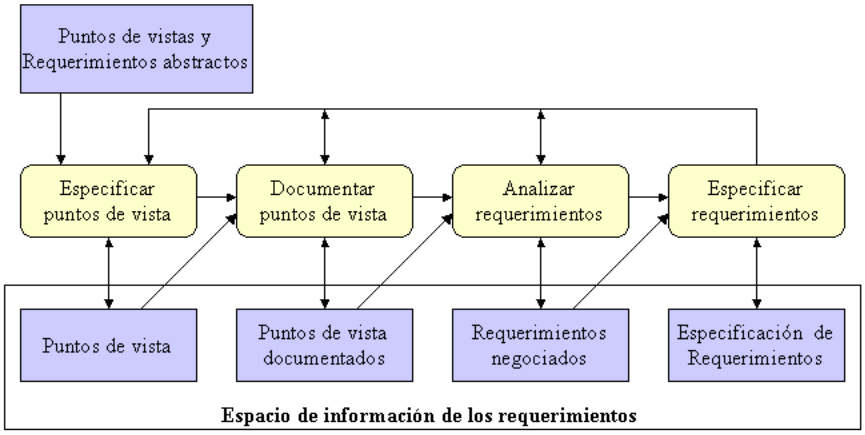


Figura 4.30 El proceso del modelo VORD.

Notación grafica para representar los punto de vista en VORD.

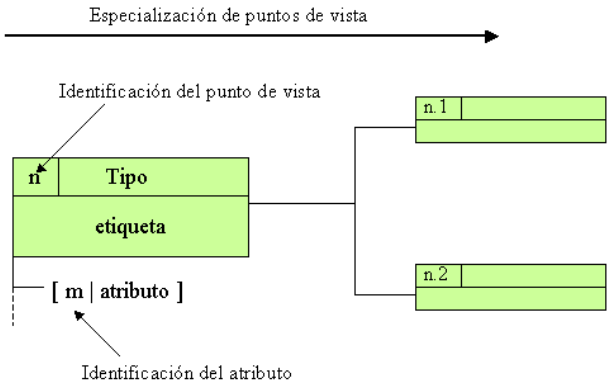


Figura 4.31 Notación para puntos de vista.

Capítulo 5

Herramientas de Software para la Administración de los Requerimientos

Cada vez es más común el uso de las herramientas de software en el desarrollo de sistemas, estas herramientas pueden integrarse y constituir una herramienta CASE compleja que puede abarcar todo el proceso de desarrollo de un sistema. En este capítulo se realiza una revisión de algunas herramientas de software para la administración de los requerimientos. Aunque todas las herramientas de administración de los requerimientos conservan un propósito común, también existen diferencias entre una y otra herramienta que resultan en ventajas y desventajas para su aplicación en los diferentes contextos de desarrollo. En general, las herramientas de software para la administración de los requerimientos, se concentran en administrar y producir el documento de especificación de los requerimientos del sistema.

5.1 VORDTool

La herramienta VordTool fue desarrollada por Ian Sommerville y Katonya en el año de 1996 [33], y se basa en la especificación de requerimientos orientado a puntos de vista. El modelo adoptado por VORDTool es orientado a servicios donde los puntos de vista son análogos a clientes en un sistema cliente-servidor. El sistema capta los servicios como puntos de vista y los puntos de vistas pasan información de control al sistema. Los puntos de vista mapean clases de usuarios finales de un sistema o las interfaces hacia otros sistemas. Los puntos de vista que constituyen la base del modelo son conocidos como puntos de vista directos. Los puntos de vista que respecta a otros sistemas que tienen influencia en el dominio de la aplicación también son

considerados y son llamados puntos de vista indirectos. Ambos puntos de vistas representan a los requerimientos funcionales y no funcionales respectivamente.

1. **Puntos de vista Directos**. Corresponden directamente a clientes que reciben servicios del sistema y envían información de datos y de control al sistema, y puede ser cualquier sistema usuario/operador u otro subsistema que interactúa con el sistema analizado.
2. ***Puntos de vista Indirectos***. Los puntos de vista indirectos tienen un interés en algunos o en todos los servicios del sistema, pero no interactúan directamente con él. Los puntos de vista indirectos pueden generar requerimientos que restringen los servicios entregados a puntos de vista directos.

El método VORD está basado en tres pasos iterativos que son:

1. **La Identificación y estructuración de los puntos de vista** trata de identificar los puntos de vista, y son declaraciones abstractas de las necesidades de la organización
2. **La Documentación de los puntos de vista** consiste en documentar los puntos de vista identificados. La documentación consiste en:
    a. Etiquetar y describir el punto de vista identificado.
    b. Rastrear los tipos de puntos de vista como directos o indirectos.
    c. Definir atributos que caracterizan a los puntos de vista en el dominio de la aplicación.
    d. Identificar los requerimientos de puntos de vista, estos incluyen un conjunto de servicios requeridos, requerimientos de control y requerimientos funcionales.
    e. Definir los escenarios de eventos que describen la interacción entre los puntos de vista y el sistema propuesto.
    f. Historial de puntos de vista que describen la evolución de los requerimientos de los puntos de vista.
3. **El Análisis y especificación de los requerimientos de puntos de vista** se refiere con la identificación de conflictos y su solución.

La herramienta VORDTool se basa en la definición de los **puntos de vistas directos** que corresponden con los requerimientos funcionales y los **puntos de vistas indirectos** que corresponden con los requerimientos no funcionales. Mediante este esquema, se definen los nodos descendientes. En los puntos de vista directos se establecen dos jerarquías, los puntos de **vista directos del operador** y los puntos de **vista directos del sistema**. En los puntos de vista

del operador, se consideran los usuarios o sistemas externos que obtienen o proporcionan un servicio al sistema analizado, y para los puntos de vista del sistema se consideran los subsistemas internos o componentes del sistema en desarrollo. La figura 5.1 ilustra la jerarquía de puntos de vista abstracta definida en VORDTool.

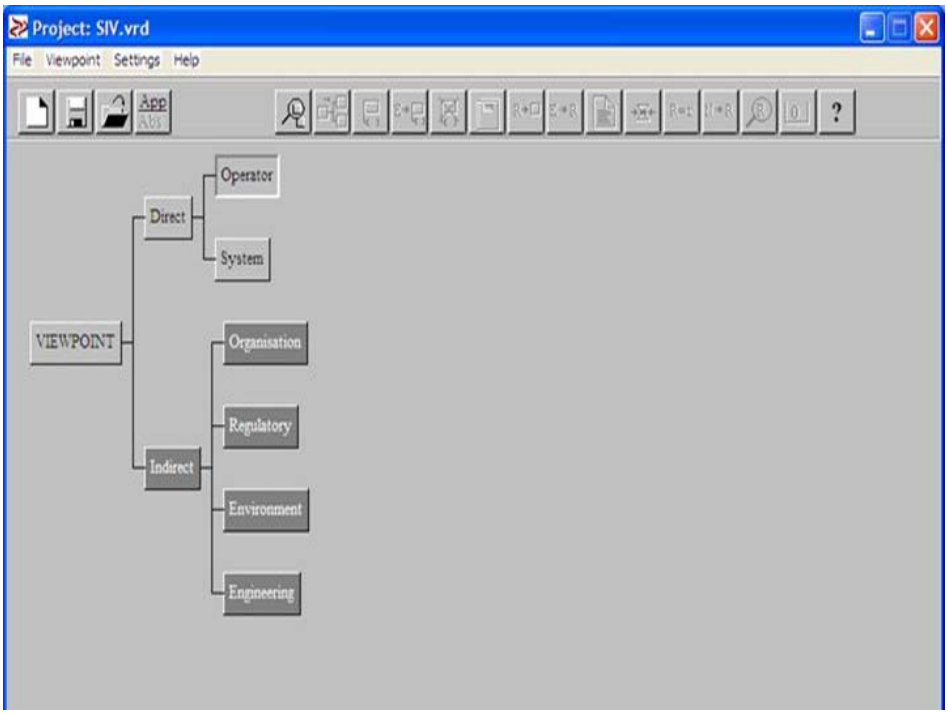


Figura 5.1 Jerarquía de Puntos de Vista en VORDTool.

Para los puntos de vista indirectos se tienen cuatro jerarquías de nodos, los puntos de vista no funcionales de la **organización** del negocio, **regulación**, **ambiente** e **ingeniería**. Estos últimos tres se refieren a requerimientos no funcionales propios del sistema de cómputo en donde residirá el sistema, y las restricciones de tecnología.

En VORDTool, para agregar un punto de vista a la jerarquía de clases abstractas, se elige el icono  , el cual mostrará una forma donde describirá el punto de vista que desee agregar, como se ilustra en la figura 5.2.

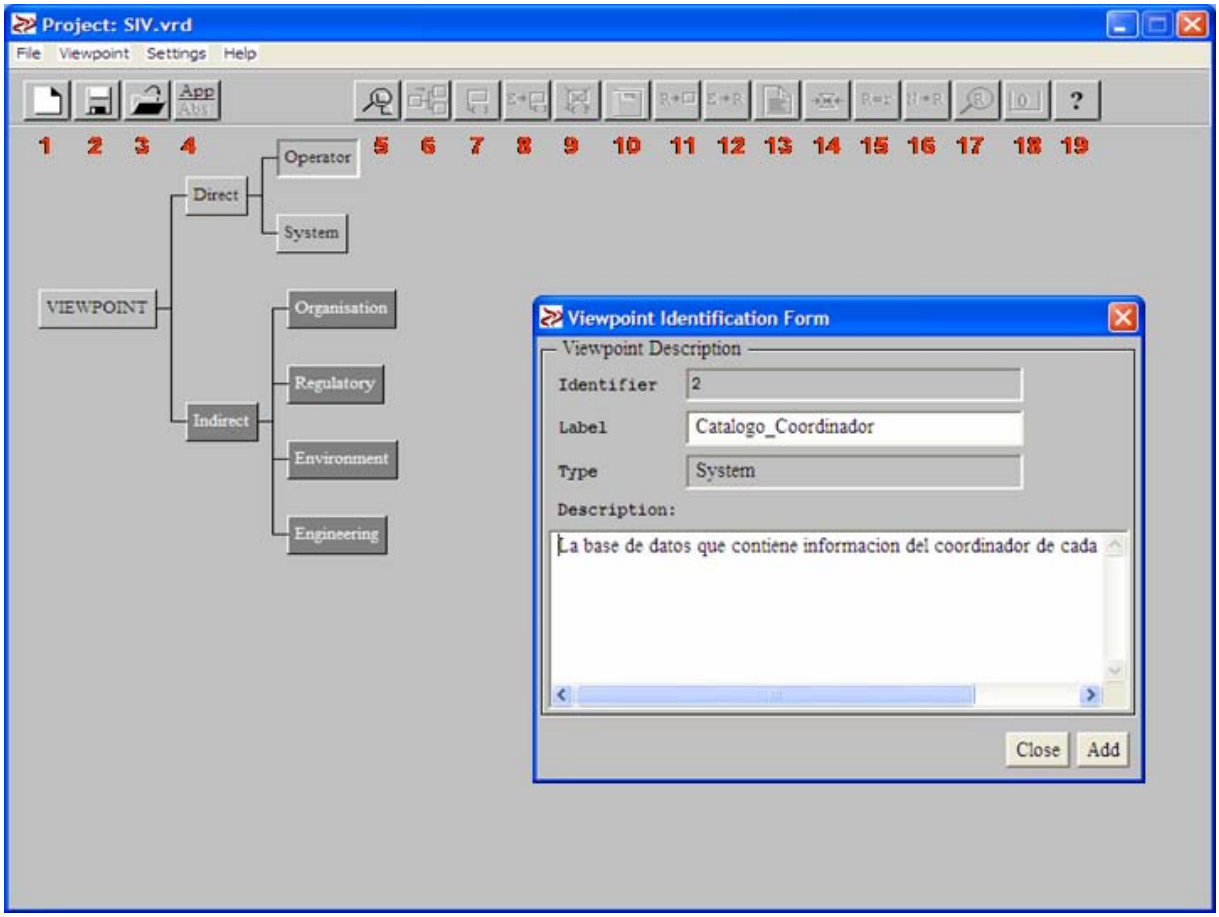


Figura 5.2 Agregando un punto de vista.

VORDTool proporciona dos maneras de mostrar el esquema de puntos de vista de sus proyectos, una es mediante el esquema jerárquico de clases abstracta (como en la figura 5.1), y la otra mediante un esquema descriptivo de los nodos o puntos de vistas descendientes en el editor canvas de la aplicación (como aparece en la figura 5.3). Se puede alternar de un esquema a otro, mediante el icono  .

Para asignar atributos a un punto de vista seleccionado, se utiliza el icono  , por ejemplo en la figura 5.3 se muestra como agregar el atributo *Num_Emp* al punto de vista *Coordinador*. Cada punto de vista definido le corresponde un identificador numérico que se muestra en la esquina superior izquierda, también los atributos tienen un identificador que depende del identificador del punto de vista al que corresponden.

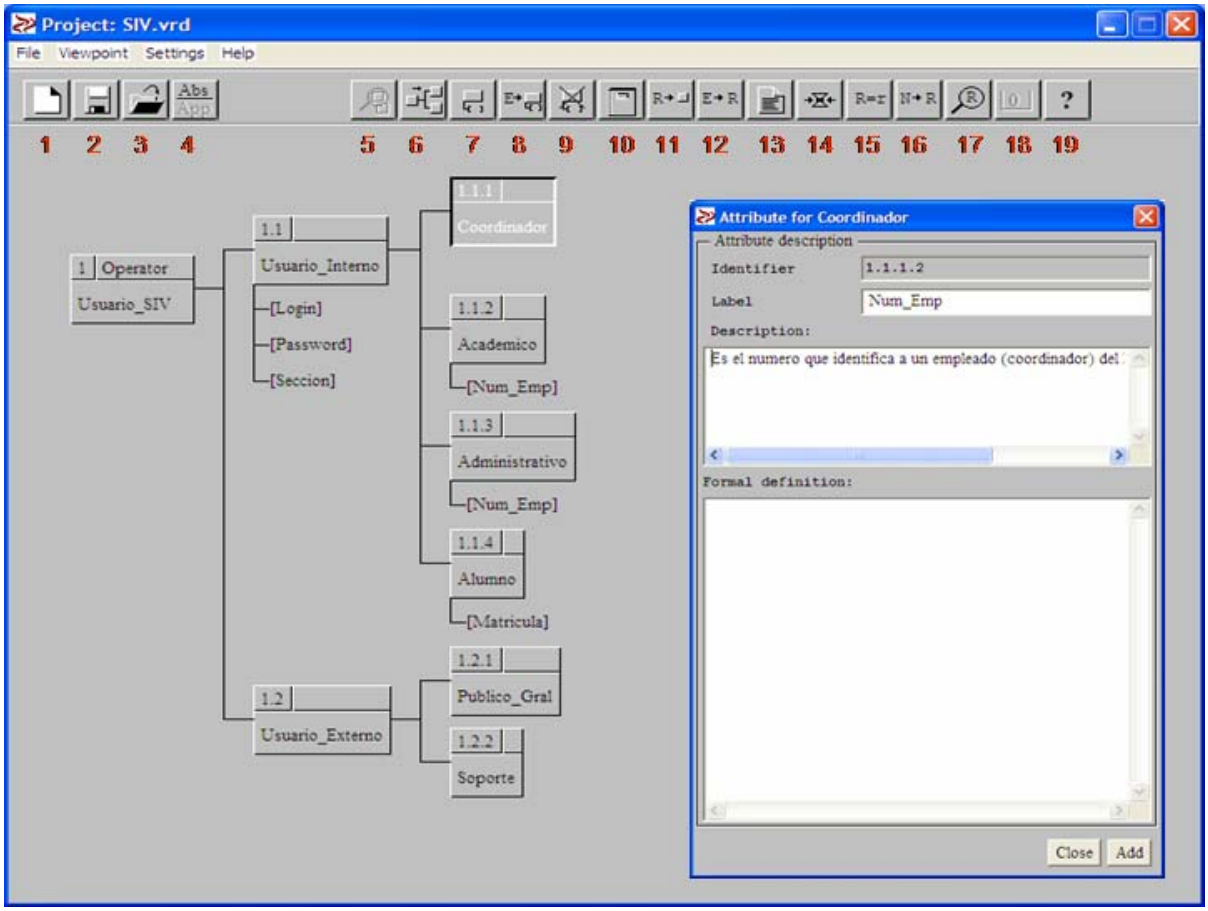


Figura 5.3 Agregando atributos a un punto de vista.

En este contexto, cada punto de vista le corresponde uno o más requerimientos que lo definen, y para agregar un requerimiento a un punto de vista, en VORDTool se utiliza el icono  . Primero seleccione el punto de vista, enseguida el icono para agregar un requerimiento, esta acción despliega una forma donde se coloca el identificador o etiqueta del requerimiento, la descripción del requerimiento, el origen y la razón de importancia en el sistema. Con la opción *Racionale*, puede agregar los siguientes datos: La prioridad del requerimiento y su justificación, el tipo de requerimiento y los requerimientos asociados. En la figura 5.4 se ilustra la pantalla de VORDTool para agregar el requerimiento *Elegir_Curso* al punto de vista *Alumno*, para el ejemplo del SIV.

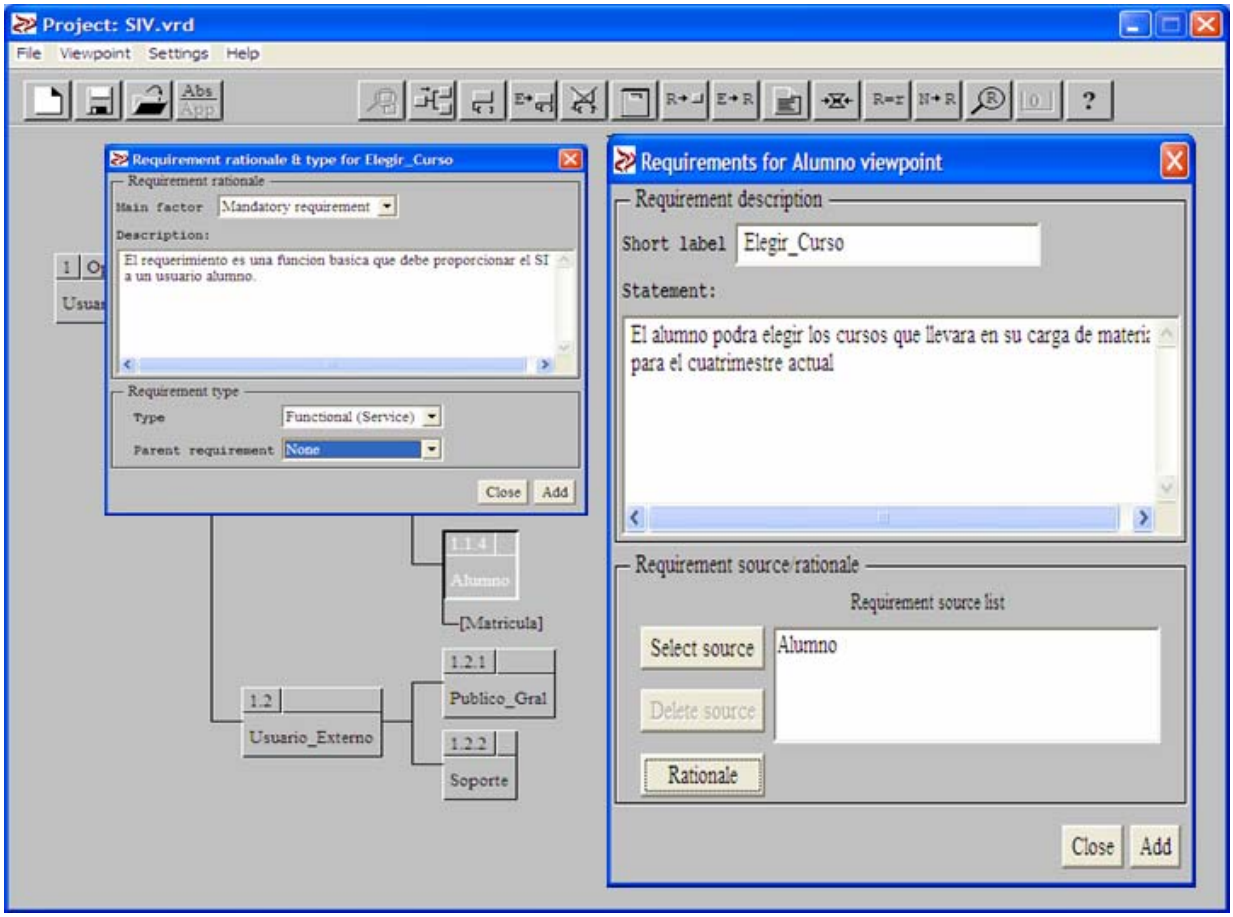


Figura 5.4 Asignar requerimientos a un punto de vistas.

Otra de las características de VORDTool es que puede describir los conflictos encontrados entre los requerimientos, mediante el icono R=r de la barra de herramienta. Para revisar los requerimientos asignados a cada punto de vista, el procedimiento es el siguiente. Se elige un par de requerimientos que se van a analizar, si se encuentran conflictos en la casilla *conflict found* se escriben recomendaciones para resolver los conflictos de los requerimientos analizados. Estas recomendaciones podrán ser utilizadas más tarde para negociar con el cliente los requerimientos en conflictos. En la figura 5.5 se muestra la ventana de la forma para identificación de conflicto para los requerimientos del ejemplo SIV, en este caso son *Ver_Curso*, y *Elegir_Curso*, ambos requerimientos no presentan conflictos.

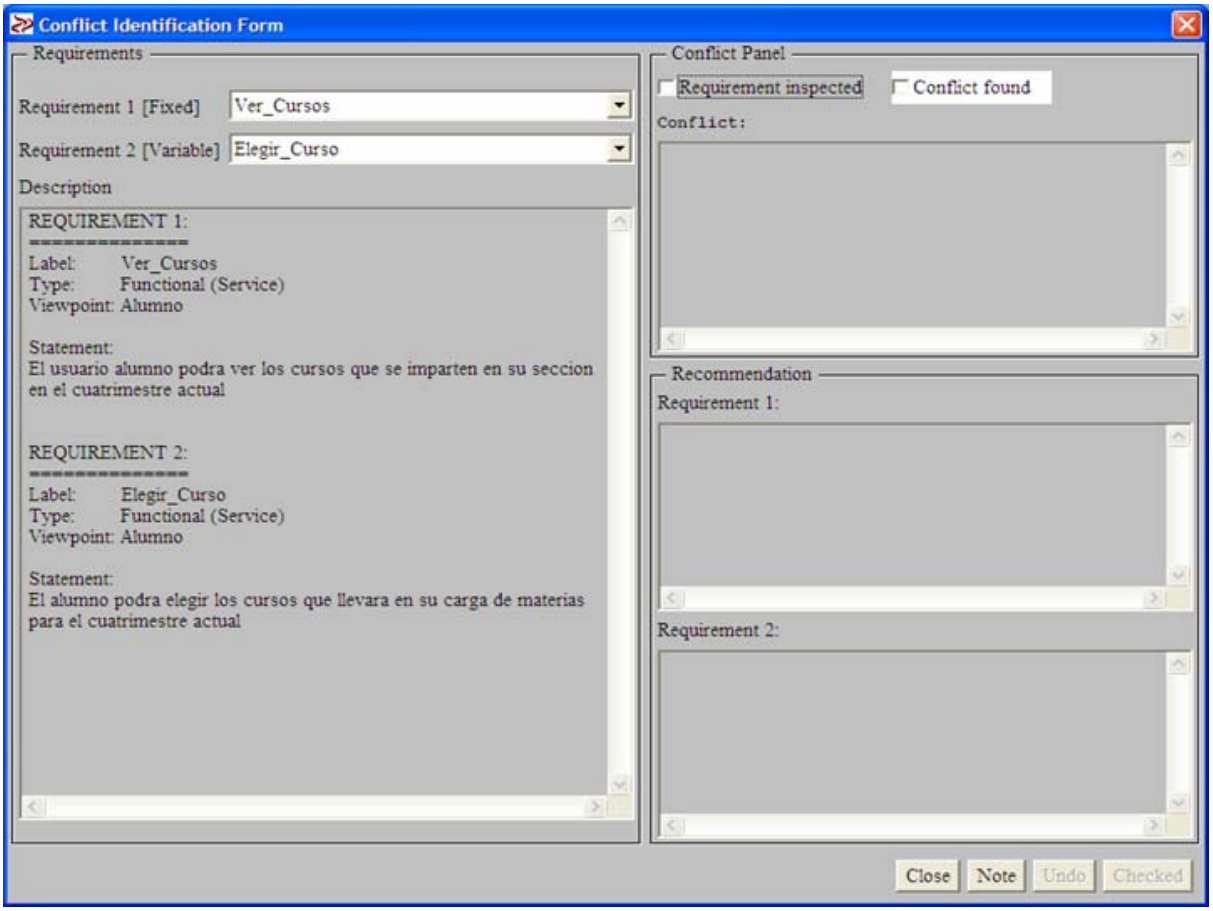


Figura 5.5 Forma de identificación de conflictos.

En VORDTool, la revisión de los requerimientos, puntos de vista y las notas puede llevarse acabo de manera automática, mediante el icono  . Esta acción proporciona una forma como la que se ilustra en la figura 5.6, donde aparecen atributos como el número de la revisión, la fecha y el título que define la revisión, el nombre de las personas que llevan a cabo la revisión, el tipo de elemento que se esta revisando, en este caso puede ser un punto de vista, un requerimiento, una nota, o un escenario de evento. De acuerdo a la selección del elemento a revisar, se activa una lista de preguntas que deben ser contestadas por las personas que realiza la revisión. En caso de que haya conflictos el revisor puede escribir sus recomendaciones en el área para aparece para este fin.

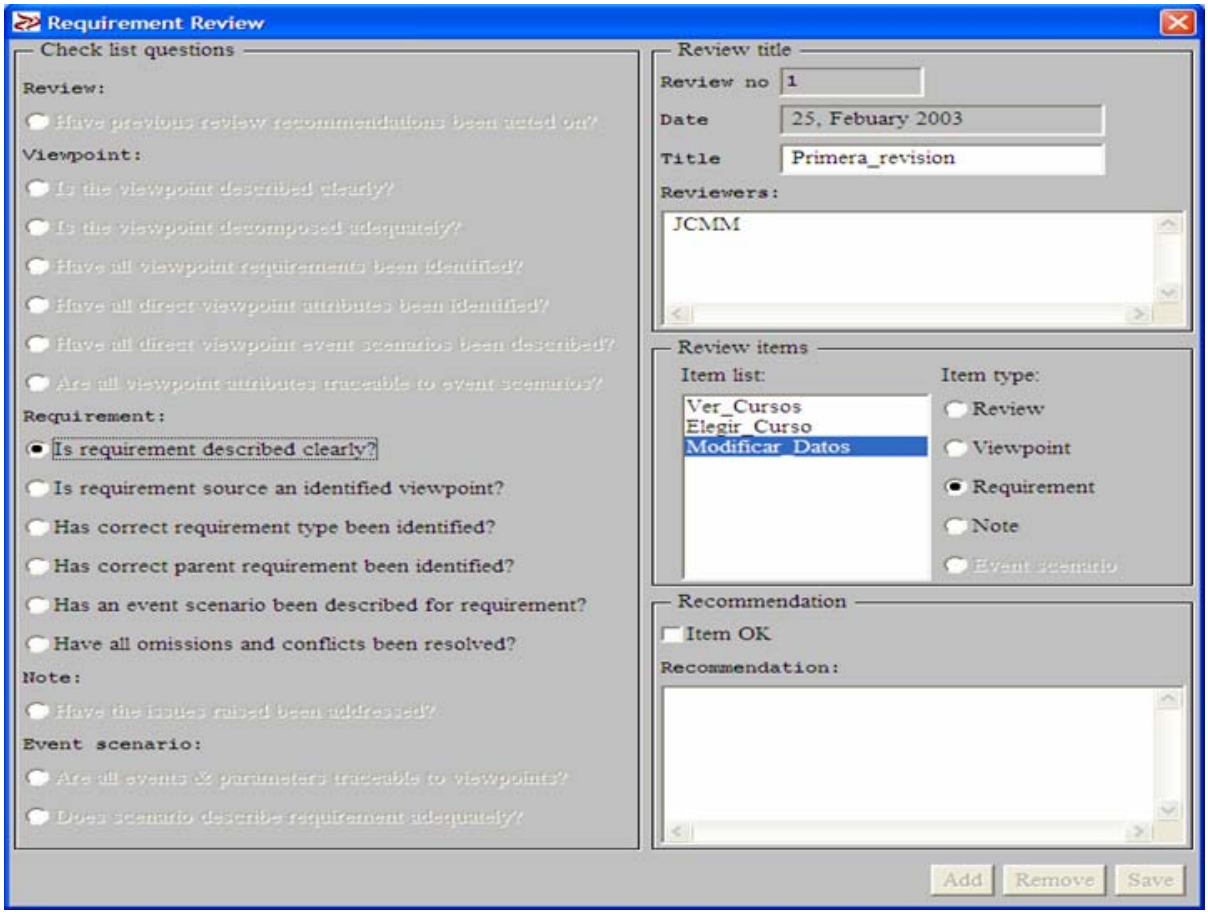


Figura 5.6 Forma para revisión de los requerimientos.

La desventaja de esta herramienta es que se encuentra ligada a un análisis orientado a puntos de vista para definir los requerimientos. Esto obliga a utilizar técnicas de obtención de requerimientos que estén orientadas a este paradigma. No posee flexibilidad para elegir un enfoque diferente como lo es la orientación a objetos mediante casos de uso.

5.2 Rational RequisitePro

Es una herramienta ofrecida por Rational Software para la administración de requerimientos [34], esta herramienta proporciona un mayor control para los requerimientos planteados por los usuarios. Mejora la administración de los cambios en los requerimientos y facilita la adición de nuevos requerimientos que surgen en este proceso. Además, con la instalación de RequisitePro en un entorno de red, permite la integración y comunicación de los miembros del equipo de desarrollo, asegurando la integridad y consistencia de los requerimientos del sistema. De esta manera se reducen los riesgos en el proyecto. Para cada nuevo proyecto, esta herramienta proporciona cinco posibles plantillas.

1. **Plantilla de Caso de Uso**. Que proporciona un enfoque utilizando la tecnología orientada a objetos (casos de uso), recomendada por Rational.
2. **Plantilla tradicional**. Esta plantilla proporciona una definición estándar de los requerimientos.
3. **Plantilla compuesta.** Es una combinación de las dos anteriores.
4. **Crear una nueva plantilla**. Para crear una nueva forma de definir los requerimientos.
5. **Plantilla en blanco**. Crear un proyecto en una plantilla sin formato.

La figura 5.7 ilustra las cinco plantillas para especificar los requerimientos, de esta forma se puede utilizar las plantillas definidas ó proponer una nueva forma de especificar los requerimientos del sistema.

Figura 5.7 Plantillas ofrecidas por RequisitePro.

Los requerimientos se encuentran documentados y organizados en paquetes. Cada paquete agrupa documentos que contienen información relacionada. En general, los paquetes pueden contener cinco tipos de documentos y son

- **Documentos en formato Word**. Contiene un **Glosario**, un **documento de Planeación**, un **documento de visión** y los **documentos de especificación de requerimientos**.
- **Vistas**. Las conforman la **Matriz de atributos**, la **Matriz de rastreabilidad** y el **árbol de rastreabilidad**.
- **Requerimientos**. Se encuentran definidos mediante los **Casos de uso**, el paquete de **Requerimientos Complementarios** y de **Sistema**.

La figura 5.8 muestra la pantalla principal de RequisitePro y las diferentes áreas que la constituyen. En el área del *explorador* se ven los diferentes paquetes que integran al proyecto.

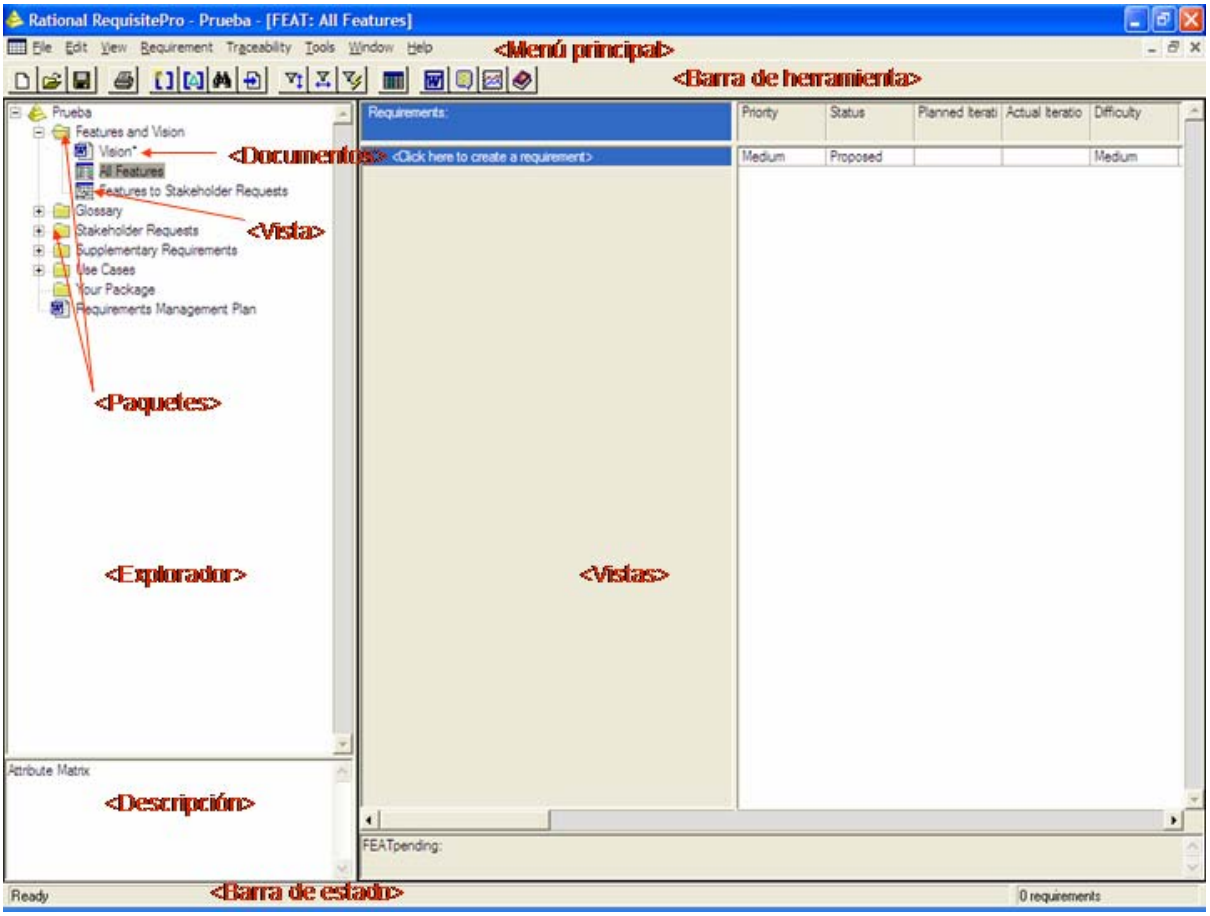


Figura 5.8 Pantalla de trabajo de Requisite Pro.

Los documentos son generados en plantillas que están definidas en *Microsoft Word*. RequisitePro utiliza un vínculo con *Word* para abrir estas plantillas. Además, añade un grupo de iconos en la barra de herramienta en la pantalla de Microsoft Word para facilitar cambios en la información de los documentos de especificación. En general, los proyectos creados en RequisitePro contienen tres tipos de documentos:

a. El **glosario** que contienen una descripción de los términos utilizados en el proyecto.
b. El **documento de planeación** que contiene una descripción del proceso de desarrollo del sistema.
c. El documento de **descripción del sistema**.
d. El documento de **definición de los requerimientos**.

La figura 5.9 ilustra la plantilla de *Word* del documento de *Vision,* en esta plantilla se describen las características del proyecto ha desarrollar. Se resaltó con un recuadro rojo, al grupo de iconos que la barra de herramienta de RequisitePro incorpora en *Word* para facilitar la manipulación de los requerimientos en los documentos de especificación.

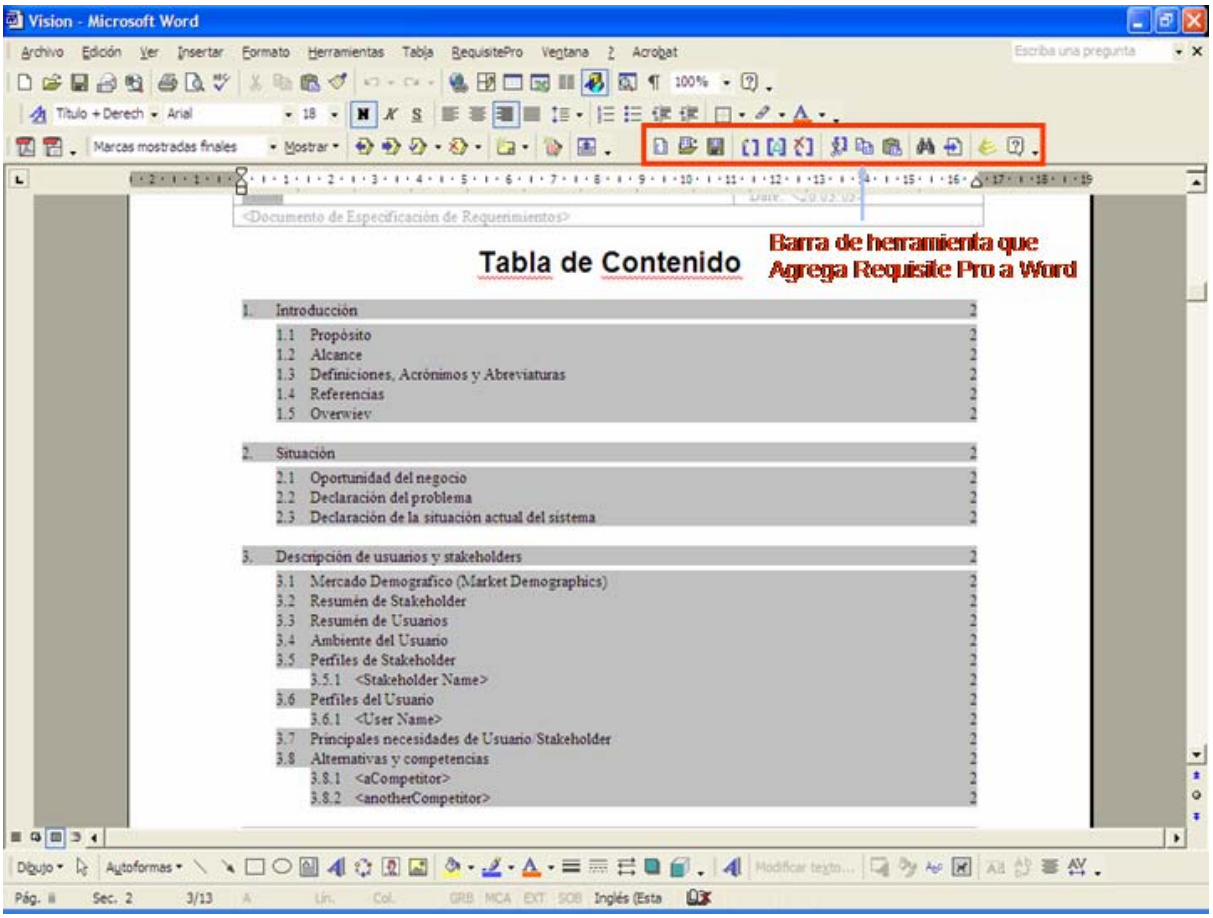


Figura 5.9 Plantilla *Vision* de RequisitePro.

En la vista de la **matriz de atributo** se encuentran especificadas características de los requerimientos. Algunas de las características mencionadas son: la prioridad de desarrollo, el estado del requerimiento, autor, dificultad de implementación, responsable, relaciones, fecha de registro, fecha de última modificación, número de versión, entre otras que ayudan a priorizar y organizar los requerimientos. En la figura 5.10 se ilustra la matriz de requerimientos con sus atributos. Los atributos de los requerimientos permiten organizar los reportes considerando algunas restricciones, por ejemplo, se pueden generar reportes ordenando los requerimientos de acuerdo a su prioridad ó dificultad de implementación.

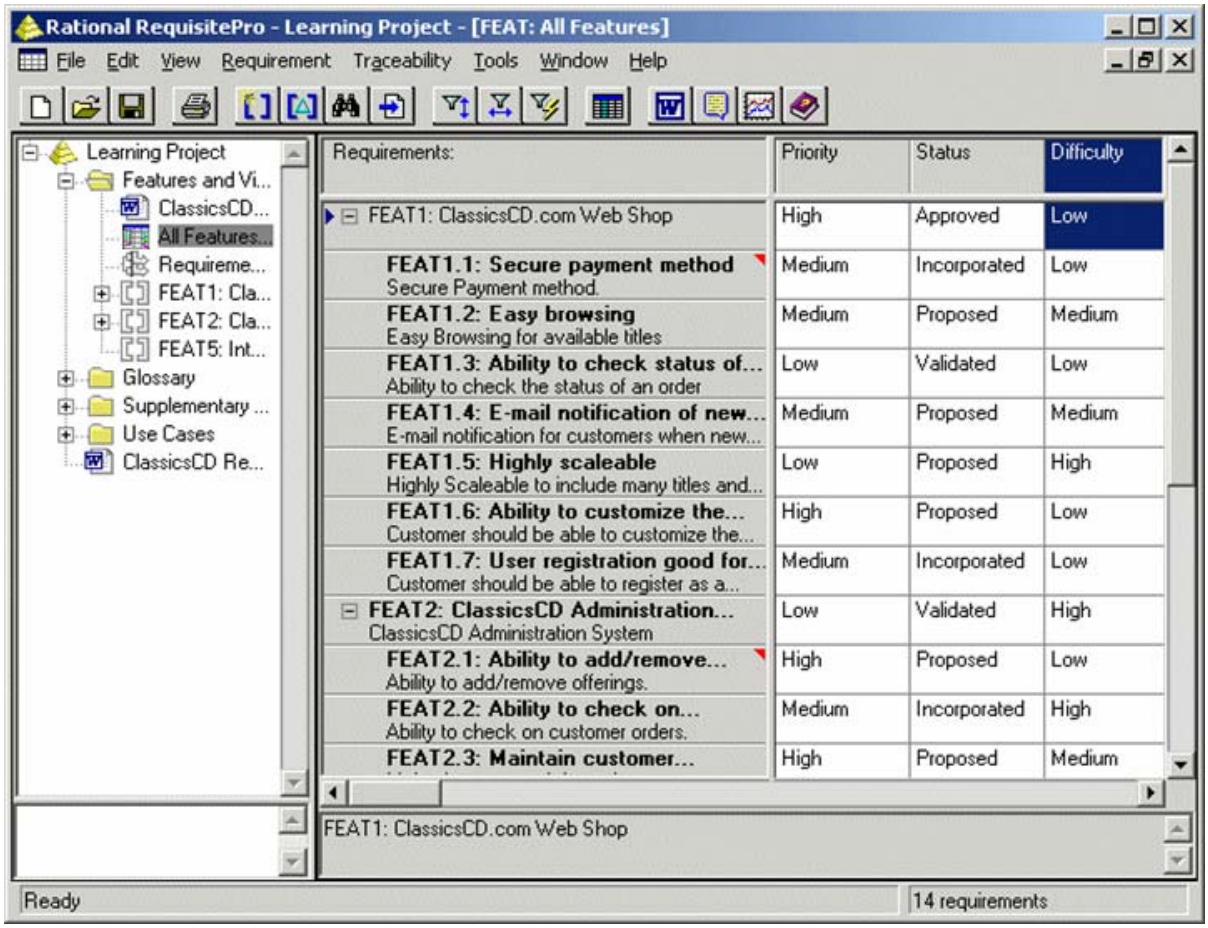


Figura 5.10 Matriz de los requerimientos o características del sistema.

En la **vista de árbol** ó **matriz de rastreabilidad** se definen las relaciones entre requerimientos, de esta manera si un cambio es realizado en alguno de los requerimientos, después de haberse establecido estas relaciones, las vistas indican visualmente mediante una flecha con una diagonal (), que estos requerimientos se encuentran en una relación bajo "sospecha". Los requerimientos se clasifican en requerimientos funcionales descritos mediante **casos de uso**, y en requerimientos **complementarios** y del **sistema** que comprenden a los requerimientos no funcionales. En la figura 5.11 se muestra una vista de árbol de los requerimientos del sistema, en este ejemplo se visualizan dos requerimientos en conflictos que son UC1.1, y sus propiedades o atributos se muestran en la parte derecha de la plantilla.

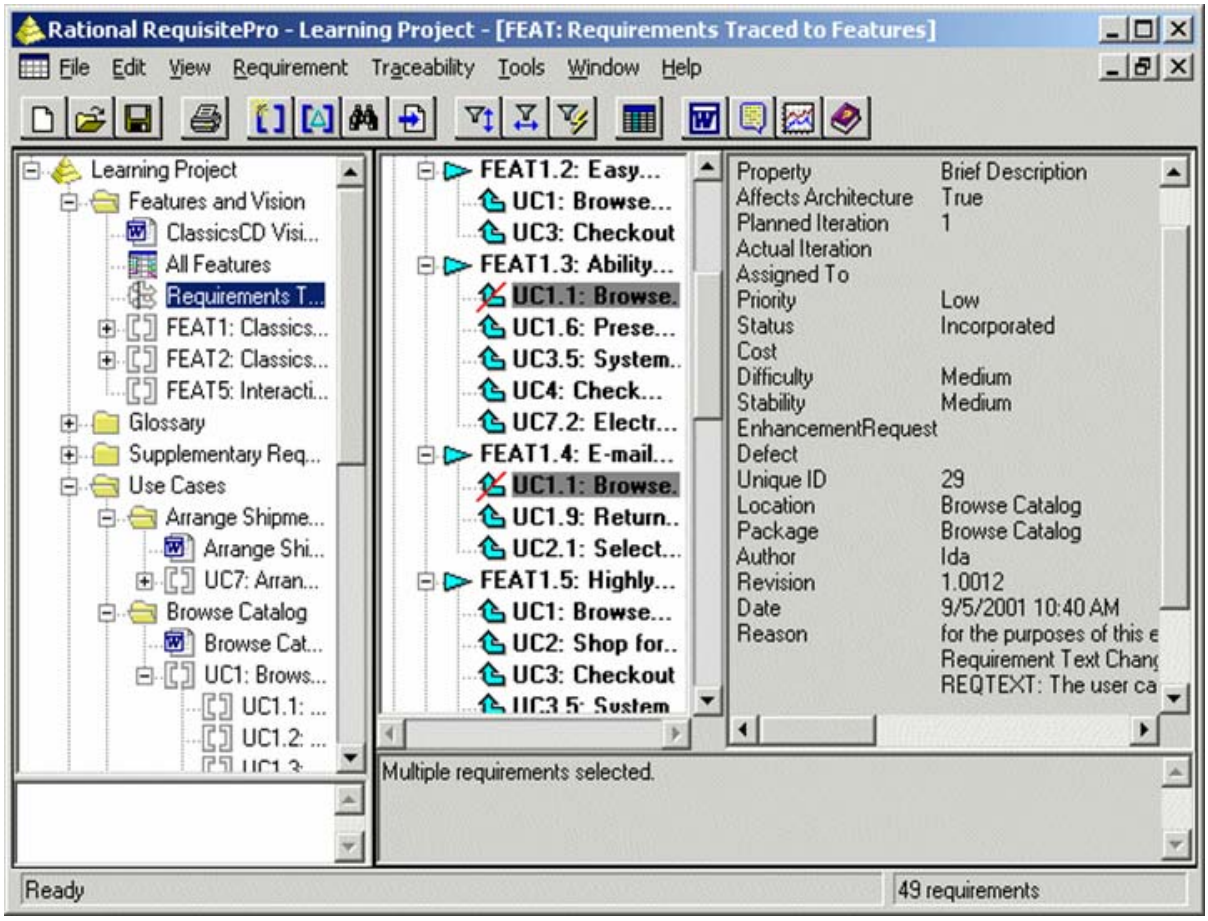


Figura 5.11 Matriz de árbol de requerimientos.

RequisitePro, también proporciona una **matriz de dependencias** entre los requerimientos. Con la matriz de dependencias se pueden revisar, asignar o eliminar las dependencias encontradas entre los requerimientos del sistema en desarrollo. Las dependencias en los requerimientos son indicadas con una flecha en la celda que muestra el traslape de ambos requerimientos ( ). Es común que los requerimientos sufran cambios en el transcurso del desarrollo del sistema, estos cambios también son reflejados en las relaciones de dependencia, al igual que en la vista de árbol, y se indican con una flecha marcada por una diagonal. La figura 5.12 ilustra la matriz de dependencia de un sistema de ejemplo.

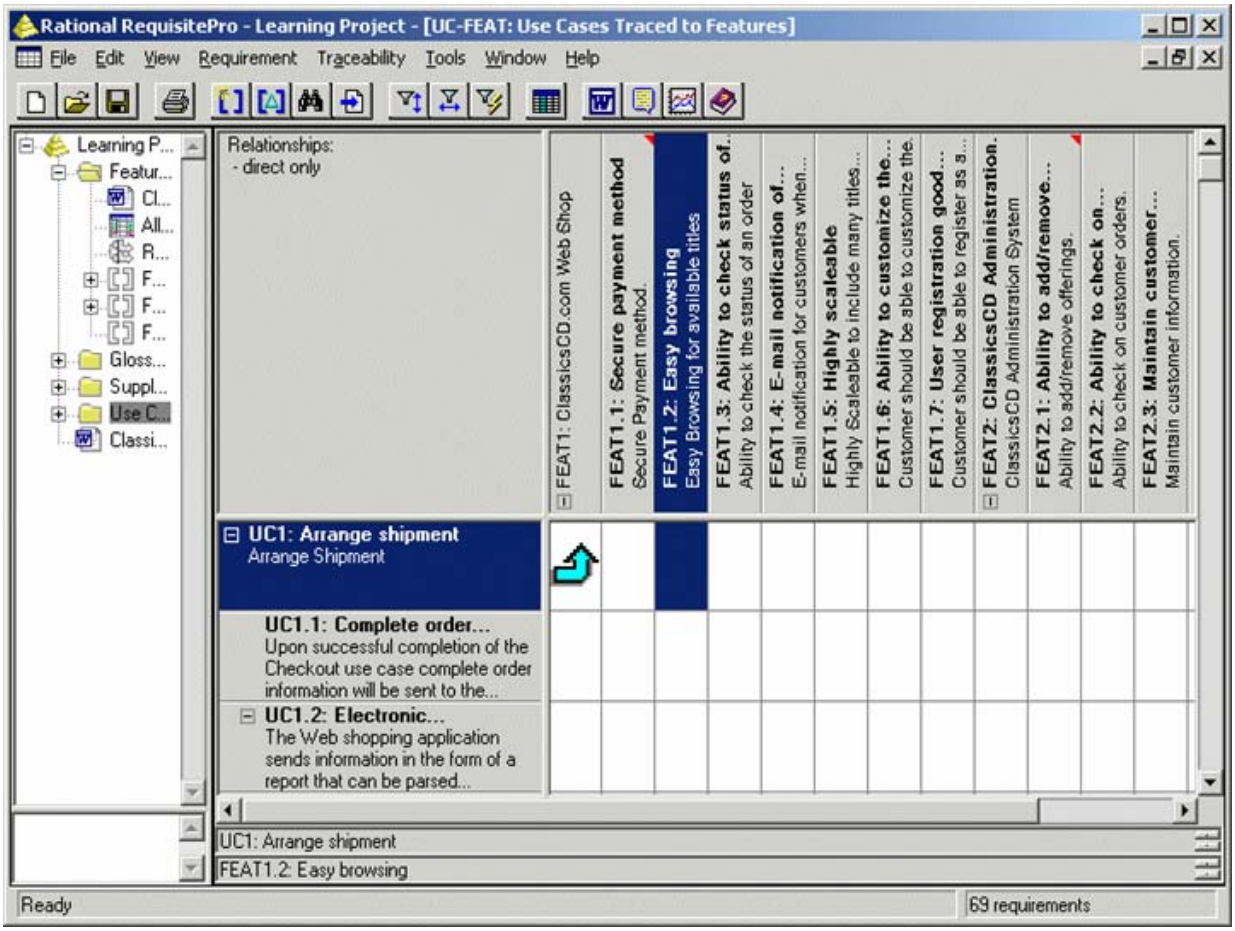


Figura 5.12 Matriz de dependencia entre requerimientos.

RequisitePro puede integrarse con otras herramientas de Software desarrolladas por Rational como son: ClearQuest, TestManagement, Suite Rational Rose for development process, que proporcionan un kit completo en el desarrollo de software.

5.3 Catalyze Enterprise

La herramienta ofrecida por **Steel Trace** [35], utiliza el paradigma orientado a objetos con un enfoque basado en Casos de Uso. Como se explica en el capítulo 3.1.5, los casos de uso proporcionan un mecanismo para capturar requerimientos funcionales de una manera comprensible y estructurada. Los casos de uso son ejecutados por actores. Un caso de uso es un hilo de funcionalidad que el sistema podría realizar, una colección de casos de uso pueden expresar el comportamiento entero del sistema. La pantalla inicial de Catalyze se muestra en la figura 5.13.

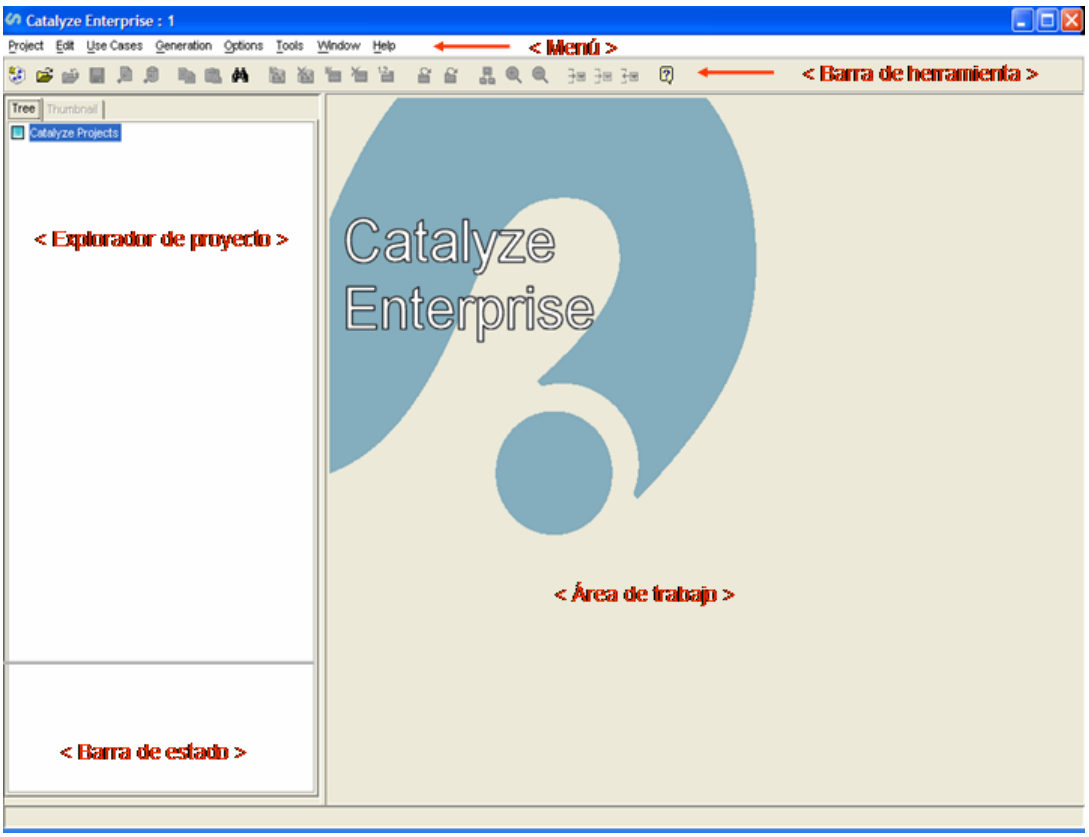


Figura 5.13 Pantalla inicial de Catalyze Enterprise.

El explorador del proyecto muestra el árbol de los componentes del proyecto, la barra de estado proporciona una pequeña descripción del elemento seleccionado, el área de trabajo describe con más detalles el elemento activo que comprende el proyecto. Un elemento puede ser una vista de casos de uso o un diagrama de flujo del sistema. La barra de herramienta presenta diferentes iconos, como el de crear un nuevo proyecto, abrir uno existente, guardar el proyecto, crear un

caso de uso, eliminarlo, moverlo, renombrarlo, mostrar diagrama de composición, generar documentos de requerimientos en Microsoft Word, actualizar el documento de requerimientos y la ayuda en línea. La información de un proyecto en Catalyze Enterprise, se encuentra organizada en la siguiente estructura:

- La **vista de árbol**. Contiene un **glosario**, una **lista de actores** y los **paquetes de casos de uso**.
- La **vista de trabajo**. Contiene la **estructura de proyecto**, los **mapas de casos de uso** y los **diagramas de flujo de los casos de uso**.
- La **vista de detalles**. La conforman una **descripción del proyecto**.

Se definen los términos usados en el proyecto mediante un **glosario**, los **casos de uso** también son detallados mediante comentarios y los **actores** se encuentran ordenados en una lista. En la figura 5.14 se muestra un ejemplo de un cajero automático de un banco.

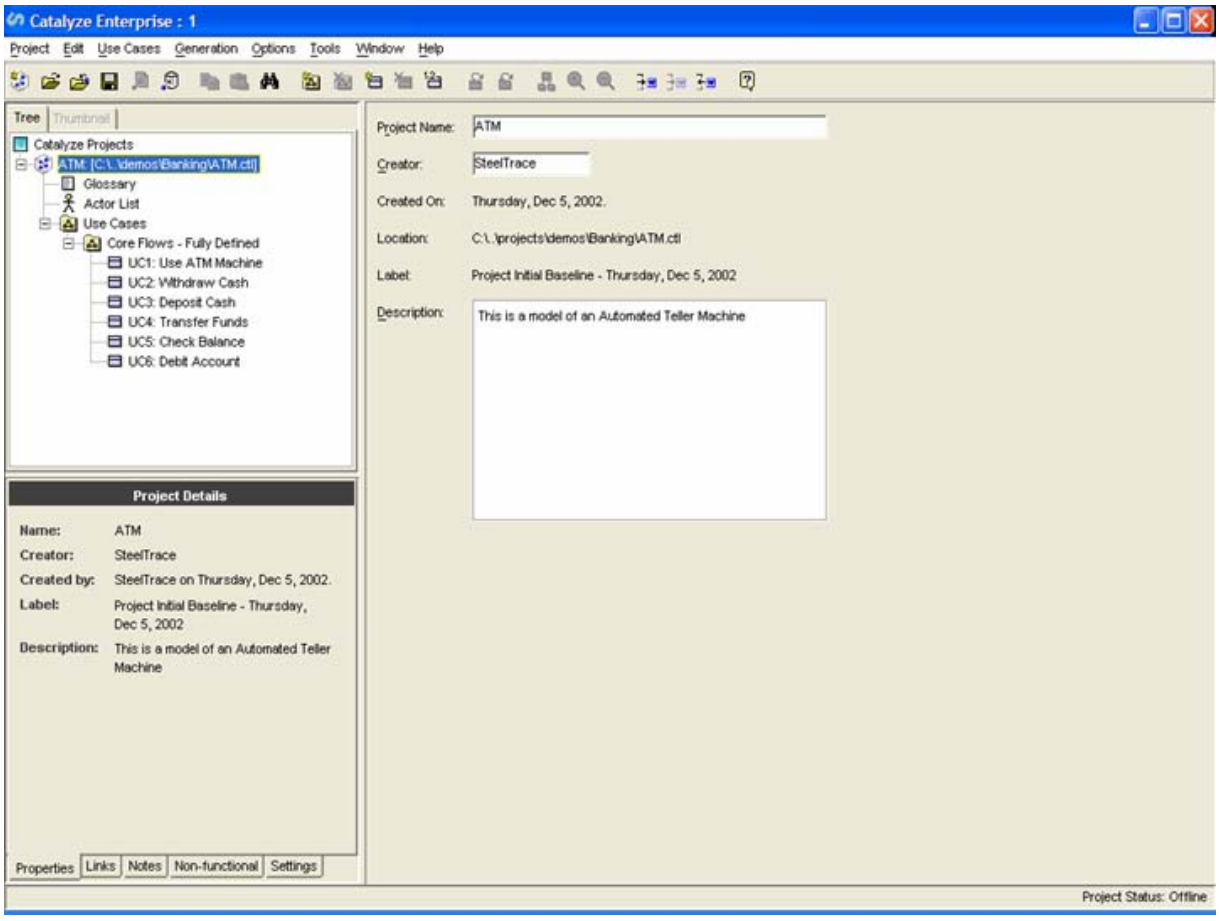


Figura 5.14 Ejemplo de un cajero automático.

La herramienta Catalize proporciona una manera sencilla de definir los términos utilizados en el proyecto, de esta forma reduce la ambigüedad para los desarrolladores y aumenta la comprensibilidad de los revisores y analistas del sistema. En la figura 5.15 se muestra una pantalla de captura del glosario para el ejemplo de un cajero automático.

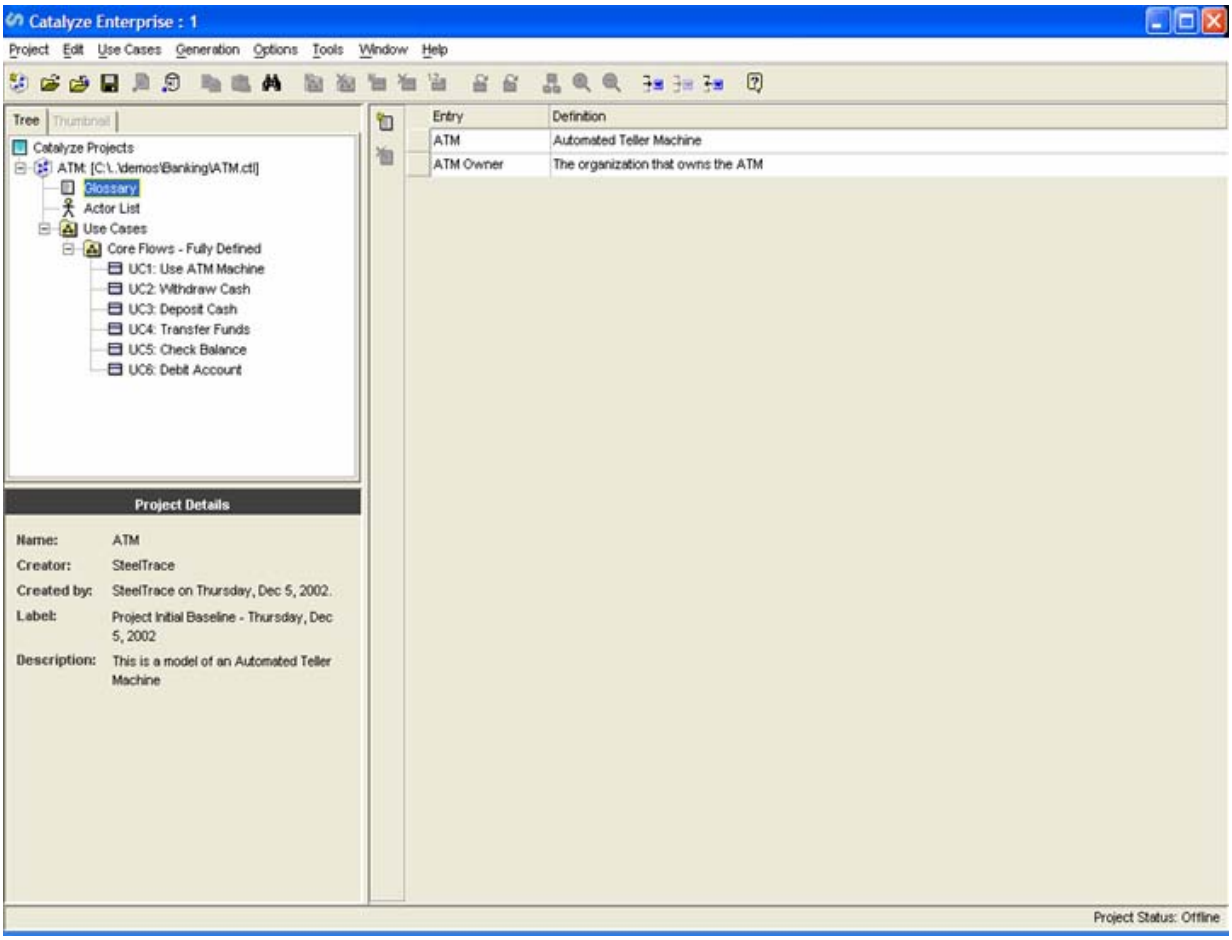


Figura 5.15 Captura del glosario del sistema de cajero automático.

También, se puede especificar una lista de actores que van a interactuar con el sistema en desarrollo, la figura 5.16 ilustra la pantalla de captura de actores para el ejemplo de un cajero automático.

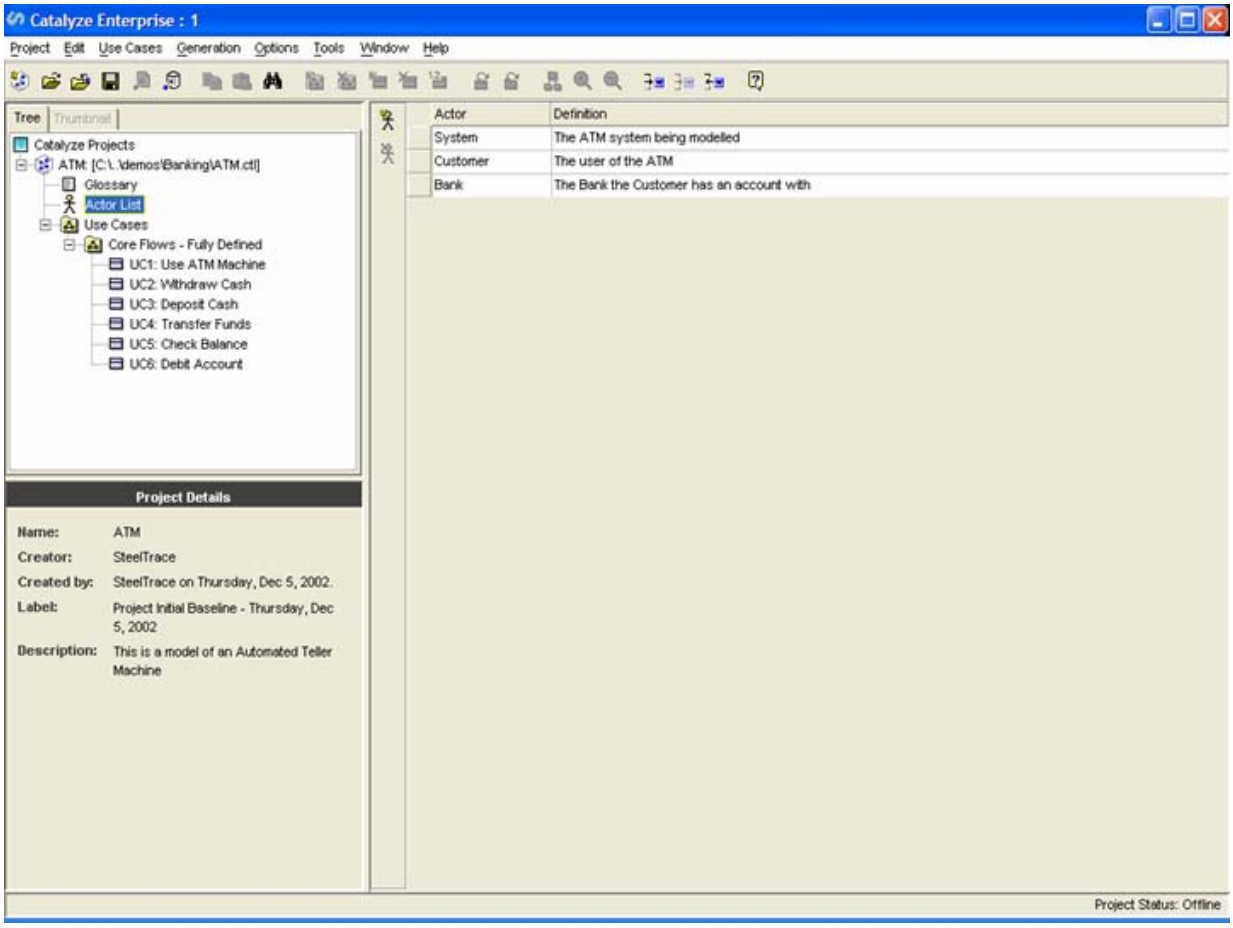


Figura 5.16 Lista de actores para el ejemplo de un cajero automático.

Los casos de uso se encuentran estructurados en **paquetes**, en donde cada paquete contiene un modelo de casos de uso, y cada **caso de uso** esta descrito mediante un escenario. Un modelo de caso de uso esta representado mediante una jerarquía descendente de casos de uso, y están ordenados de acuerdo a su participación en el sistema. La figura 5.17 y 5.18 detallan esta estructura de los casos de uso en el ejemplo anterior del cajero automático.

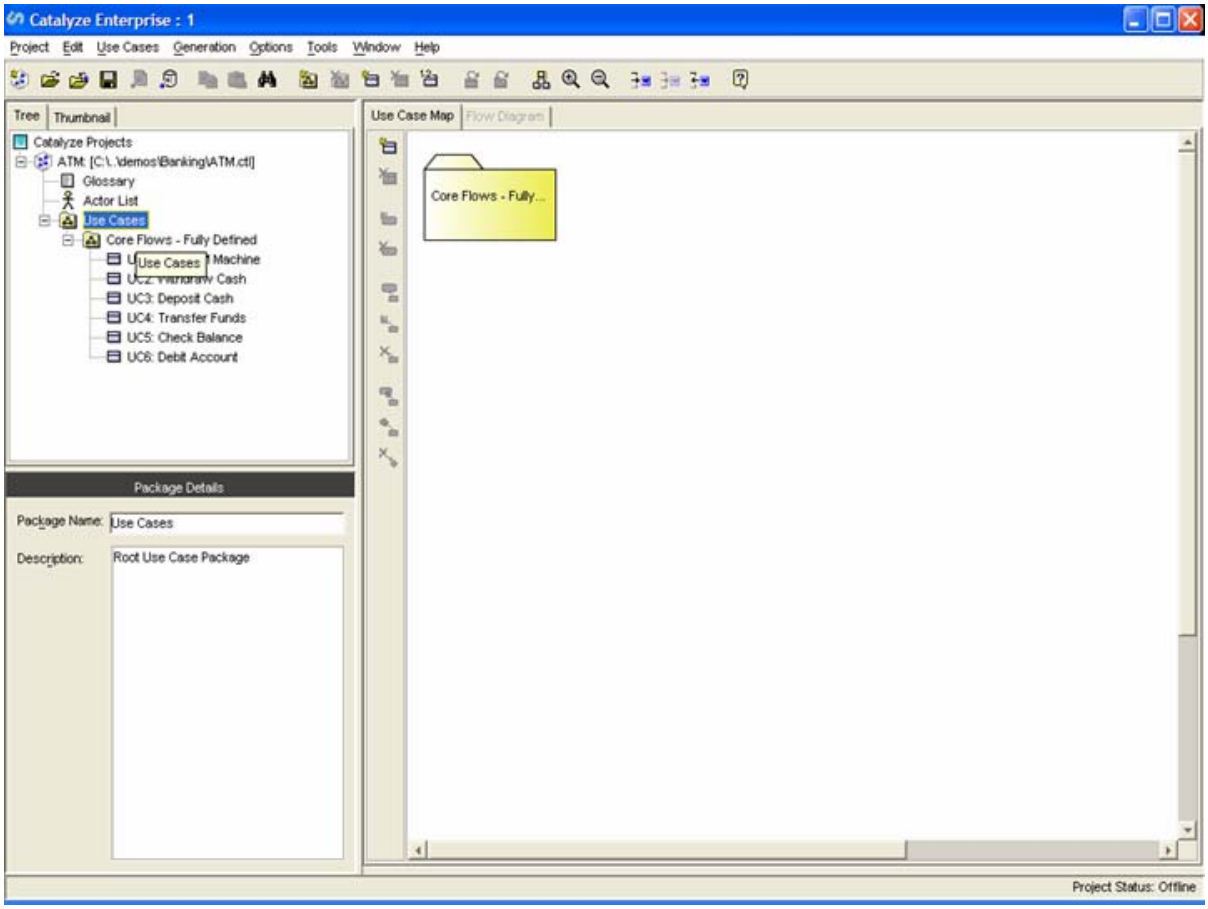


Figura 5.17 Paquete de casos de uso del ejemplo de un cajero automático.

En el ejemplo del cajero automático, el paquete mostrado en la figura 5.18 llamado *Use Cases* contiene seis casos de uso que se encuentran listados en el **árbol** de navegación (vease la figura 5.17), y en la parte derecha de la misma figura o el área de trabajo se muestra una carpeta que contiene el diagrama de estos casos de uso. La figura 5.18 ilustra el esquema jerárquico de los casos de uso que intervienen en el sistema de ejemplo de un cajero automático.

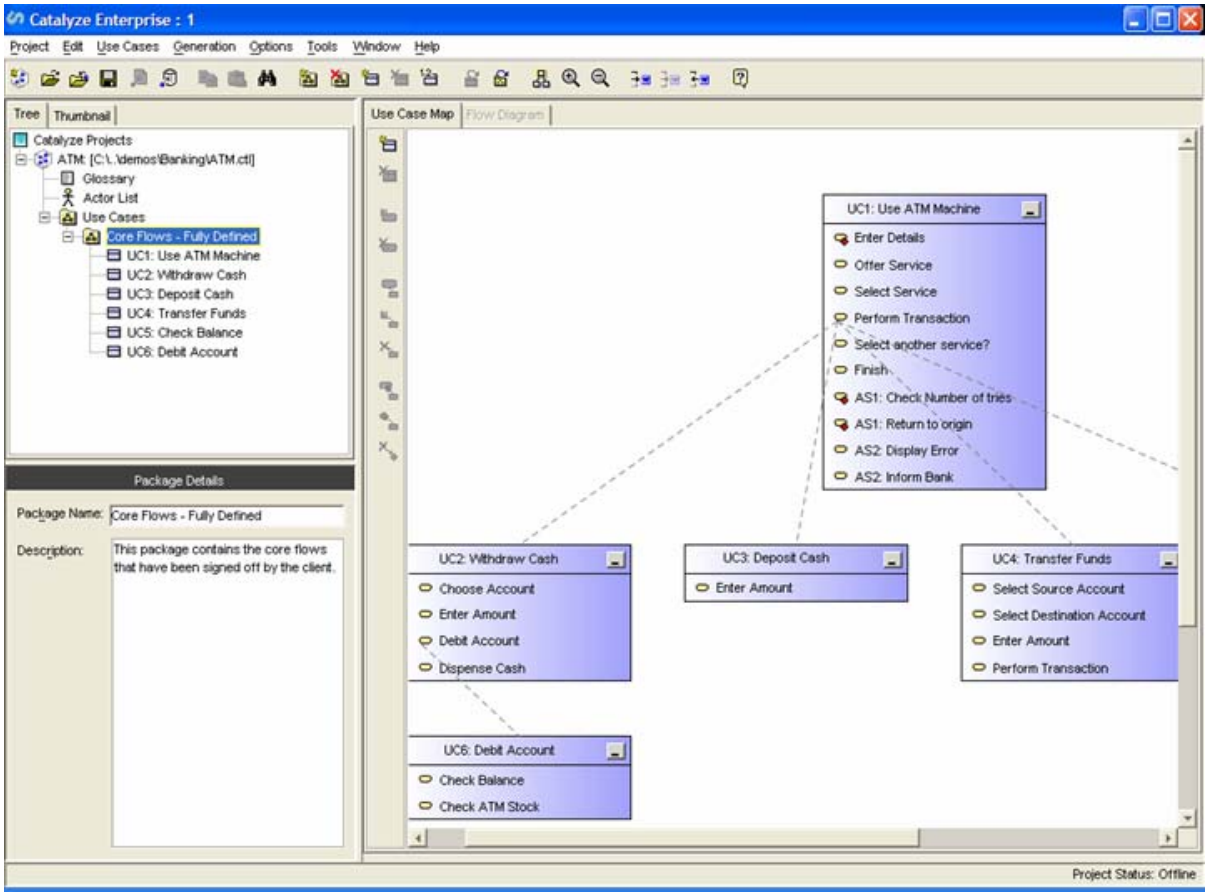


Figura 5.18 Jerarquía de casos de uso para el sistema de un cajero automático.

Cada caso de uso tiene una descripción de los actores que participan, las acciones que se ejecutan, los casos de uso que intervienen y con los que se relacionan. Los detalles de los casos de uso pueden ser editados directamente en el área de trabajo. En la figura 5.19 se muestran los detalles para un caso de uso del ejemplo del cajero automático.

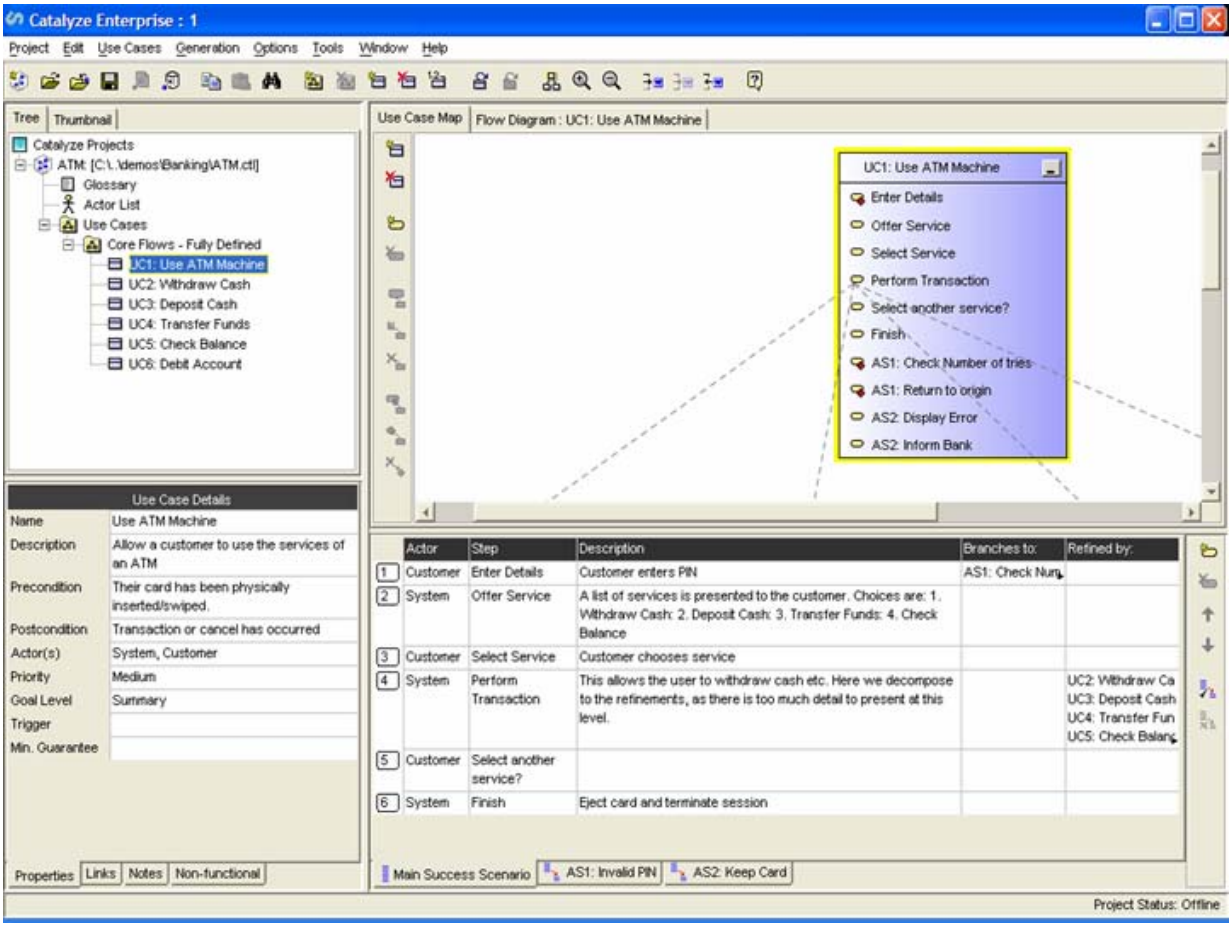


Figura 5.19 Detalles de un caso de uso para el ejemplo del cajero automático.

El caso de uso mostrado en la figura 5.19 es el *Use ATM Machine*, el cual contiene una descripción de los seis actores que participan en él, los pasos que cada actor ejecuta y los casos de uso que lo detallan. En la **vista de estado** se muestra también una descripción que contiene el nombre del caso de uso, su definición, las precondiciones y poscondiciones, los nombres de los actores que intervienen, la prioridad del caso de uso, el nivel de detalle, la acción que lo dispara y la garantía que se proporciona en su ejecución.

También se puede observar el diagrama de flujo de los casos de uso del sistema en desarrollo, lo cual permite ver como se desarrolla el flujo de secuencia del sistema. La figura 5.20 muestra la jerarquía del diagrama de flujo para los casos de uso en el ejemplo del cajero automático.

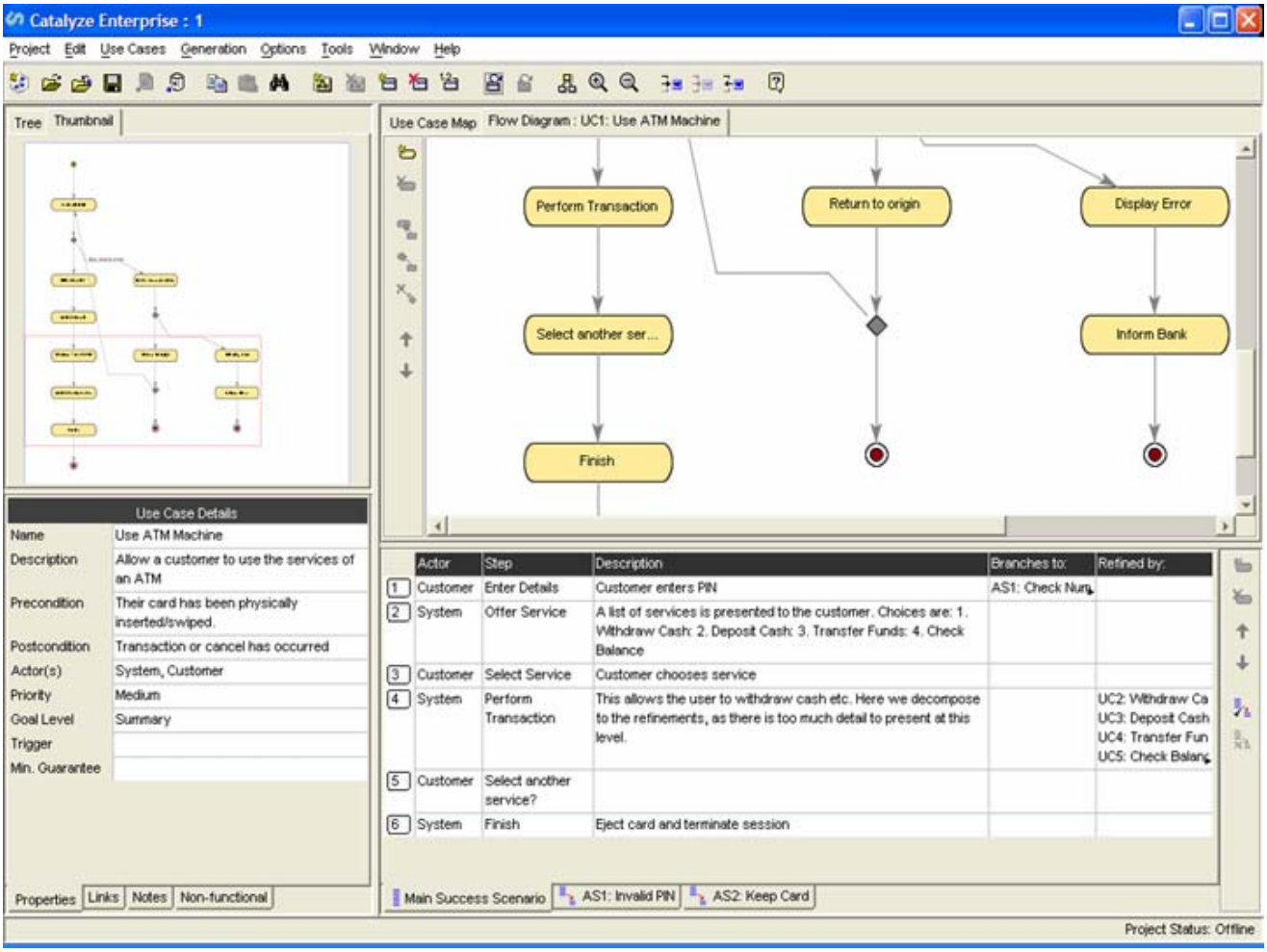


Figura 5.20 Diagrama de flujo de los casos de uso en el ejemplo de un cajero automático.

Al igual que Requisite Pro, esta herramienta utiliza un enlace con *Microsoft Word* para generar los documentos de requerimientos del sistema. Los documentos pueden generarse automáticamente a partir de la definición de los casos de uso y del documento de requerimientos se puede obtener la definición de los casos de uso.

Una de las desventajas de esta herramienta es que carece de un mecanismo de verificación de dependencias entre los casos de uso como lo hace RequisitePro con la matriz de dependencia, además que no proporciona un enlace directo hacia otras herramientas CASE.

5.4 Integral Requisite Analizer (IRqA)

El IRqA es una herramienta de software desarrollada por ***TCP Sistemas e Ingeniería*** [36]. Esta herramienta es especialmente diseñada para dar soporte a los analistas y a los ingenieros de software en el proceso de ingeniería de requerimientos. Proporciona la funcionalidad no solo de administrar los requerimientos y producir una especificación, sino también la posibilidad de formular estos requerimientos dentro del contexto del sistema. IRqA proporciona dos modos de abrir los proyectos, una consiste en abrir un proyecto existente y la otra es crear un nuevo proyecto, como se muestra en la figura 5.21.

Figura 5.21 Pantalla de apertura de proyectos en IRqA.

Los proyectos de IRqA pueden estar basados en dos tipos de enfoques de análisis:

1. **El Enfoque Funcional**. Usa técnicas funcionales para definir el modelo conceptual del proyecto.
2. **El Enfoque Orientado a Objetos**. Usa técnicas orientadas a objetos para definir el modelo conceptual del proyecto.

En la figura 5.22 se ilustra el enfoque funcional y el orientado a objetos. Se pide el tipo de usuario mediante un login y password, esto se debe a la seguridad y autorización para modificar los documentos de los requerimientos o algún elemento del sistema.

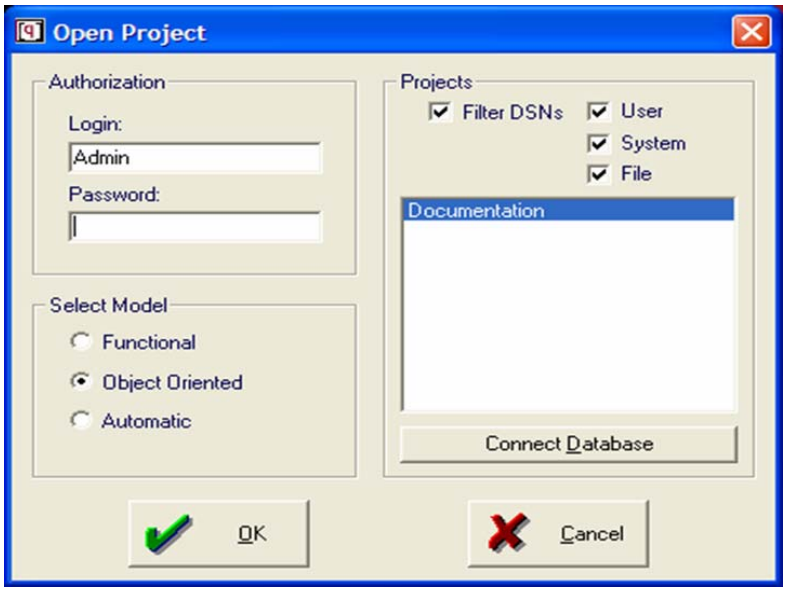


Figura 5.22 Enfoques del proyecto de requerimientos en IRqA.

La estructura general de las vistas textuales de la herramienta IRqA esta constituida por tres partes principales que son la **barra de herramientas**, el **árbol de navegación** y la **vista de propiedades**. En la barra de herramienta se pueden encontrar botones de edición, de rápido acceso, de mantenimiento, de control de versiones, de dominio y de bloques. Cada una de las vistas en IRqA contiene sólo un tipo de elementos que constituyen el proyecto. Los elementos que constituyen un proyecto en IRqA son: *requerimientos, servicios, eventos, actores, escenarios, conceptos, diagramas de casos de uso, reportes, matriz de rastreabilidad, administración de usuarios y diagramas de contexto*.

El **árbol de navegación** contiene todos los elementos que constituyen a un tipo de elemento en particular del proyecto abierto. En la vista o tabla de propiedades muestra todas las propiedades del elemento que se encuentre seleccionado. En la figura 5.23 se ilustran las vistas que constituyen a la pantalla principal de IRqA.

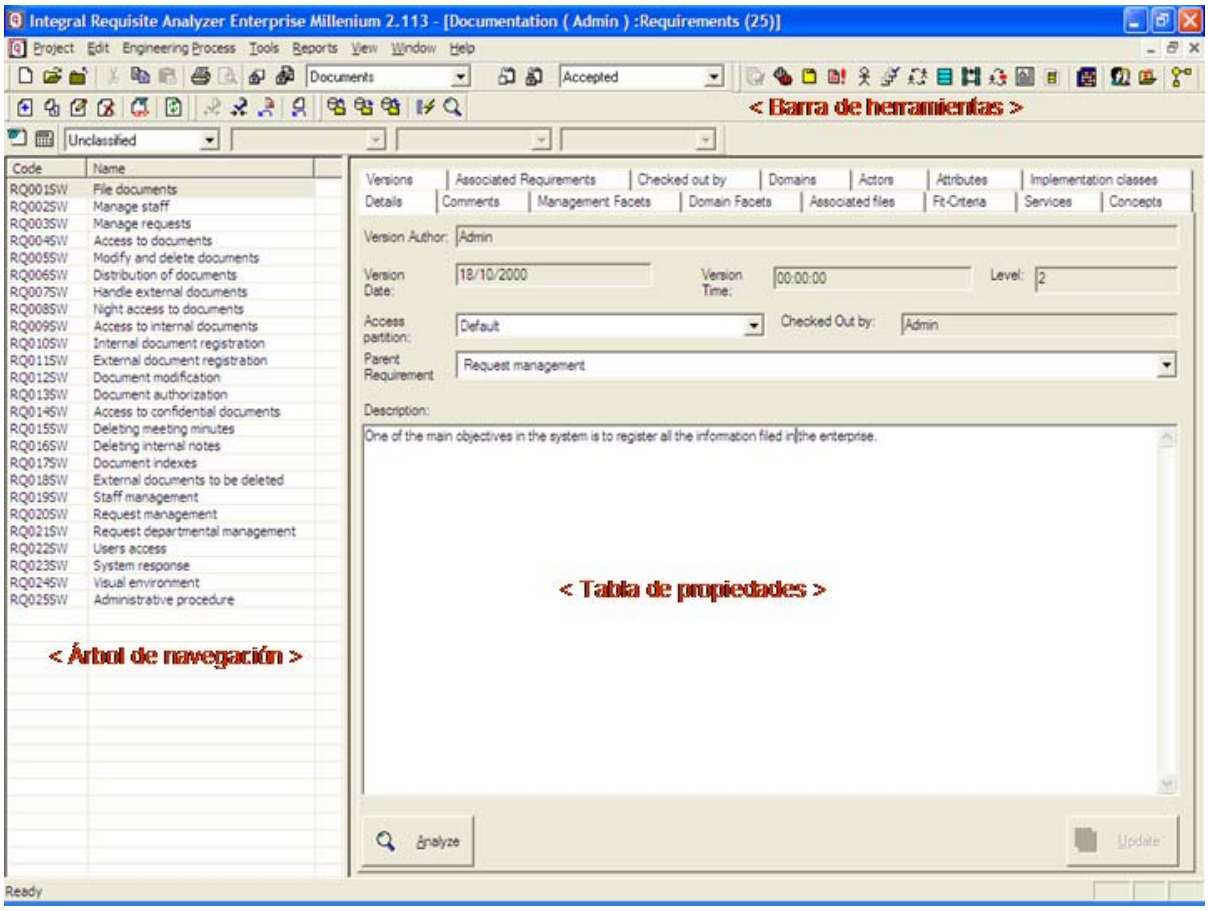


Figura 5.23 Vistas de la estructura de una pantalla en IRqA.

En la figura anterior se puede observar que en el área del árbol de navegación se muestra una lista de *requerimientos* que contienen un código y un nombre. Estos requerimientos constituyen al proyecto de ejemplo, el cual consiste en un sistema de documentación. Cada requerimiento está constituido por un grupo de atributos que se pueden observar en la vista de propiedades. Entre los atributos que más destacan son: autor, versión por fecha y hora, nivel, requerimiento padre, verificado por, y una definición del requerimiento.

Los **servicios** al igual que los requerimientos son parte de los elementos que constituyen un proyecto en IRqA, y de igual forma se muestra en una lista que aparece en el árbol de navegación, sus atributos se muestran en la vista de propiedades y pueden ser diferentes a los atributos de los requerimientos. La figura 5.24 contiene la vista de los servicios proporcionados por el sistema de administración de documentos que se utiliza como ejemplo.

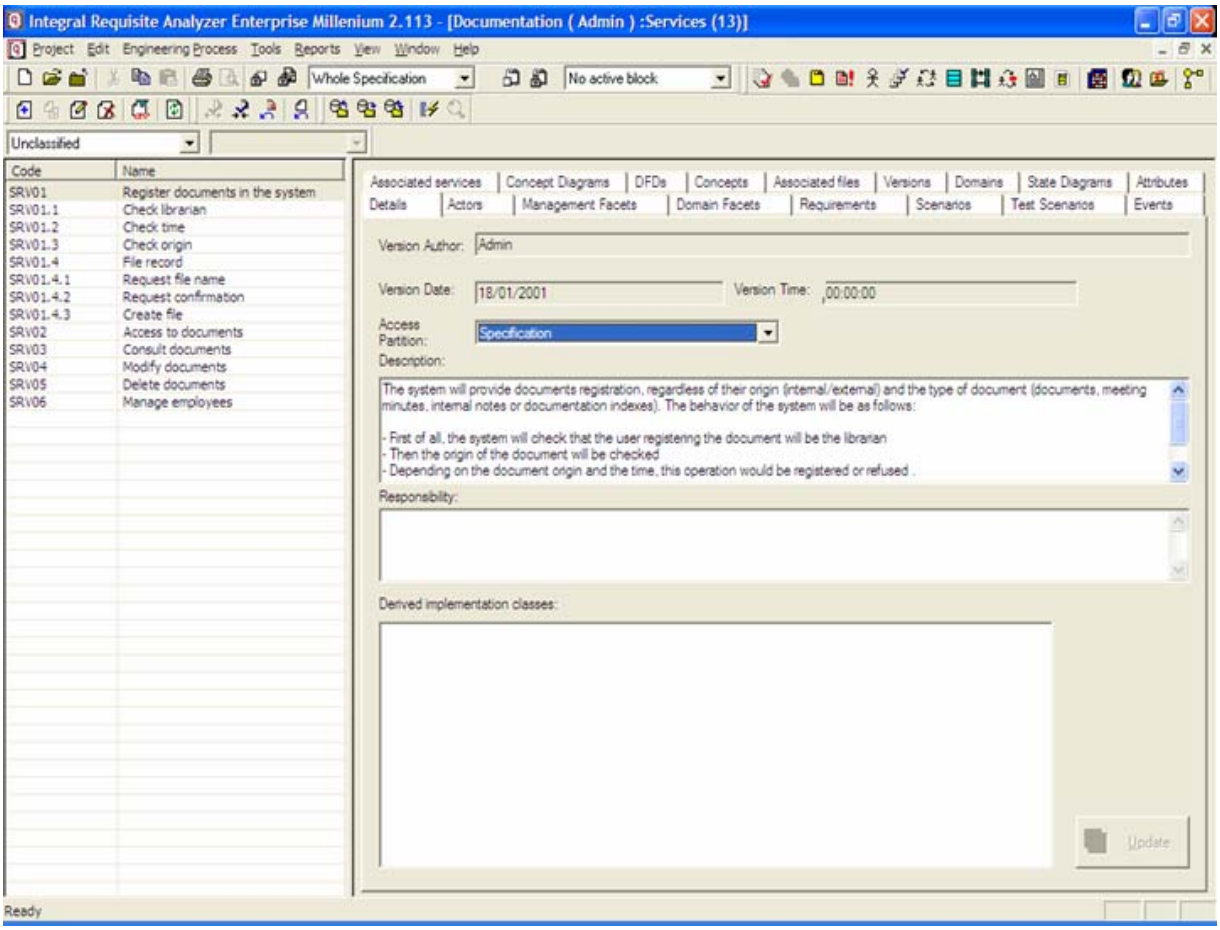


Figura 5.24 Vista de los servicios que proporciona un sistema de documentación.

En el enfoque orientado a objeto **los actores** constituyen una parte fundamental en la descripción de los escenarios y casos de uso. La herramienta IRqA los incluye como parte fundamental de una descripción del proyecto. El ejemplo que se ha utilizado, consiste en un sistema de documentación. La figura 5.25 ilustra una lista de actores que intervienen en este sistema, y son: el bibliotecario, la secretaria y el usuario. La vista de propiedades para un elemento *actor*, esta constituida por una descripción del actor, servicios asociados, requerimientos asociados, versión, dominios y atributos entre otros.

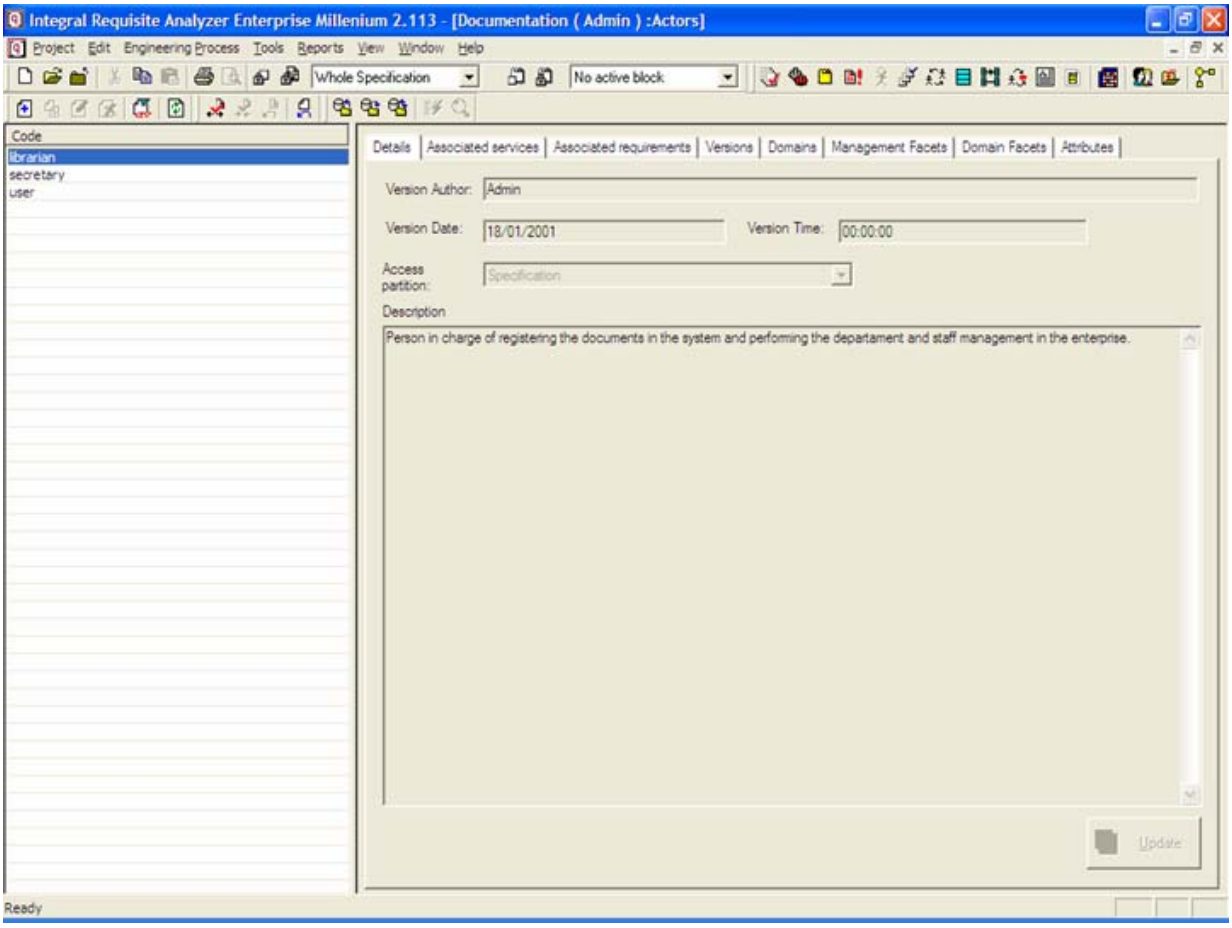


Figura 5.25 Vista de actores que intervienen en el sistema de ejemplo.

Los **escenarios** también constituyen parte importante en los requerimientos por lo que son tratados en un proyecto de IRqA. La descripción que se utiliza es mediante un identificador para cada escenario, el actor que interactúa con el sistema, los eventos que son llevados a cabo, la dirección o sentido en que se ejecutan, estos pueden ir del actor hacia el sistema, del sistema hacia al actor o en ambos sentidos. Además de otros atributos, se definen también las precondiciones y poscondiciones para que se realice el escenario, los servicios asociados, los dominios, facetas de administración, facetas de dominio y atributos.

La herramienta también proporciona un conjunto de acciones que se pueden realizar en torno a los escenarios, como agregar, editar, eliminar, mover hacia arriba o hacia abajo un escenario. En la figura 5.26 se indica la pantalla de vista de los escenarios del ejemplo de un sistema de administración de documentos.

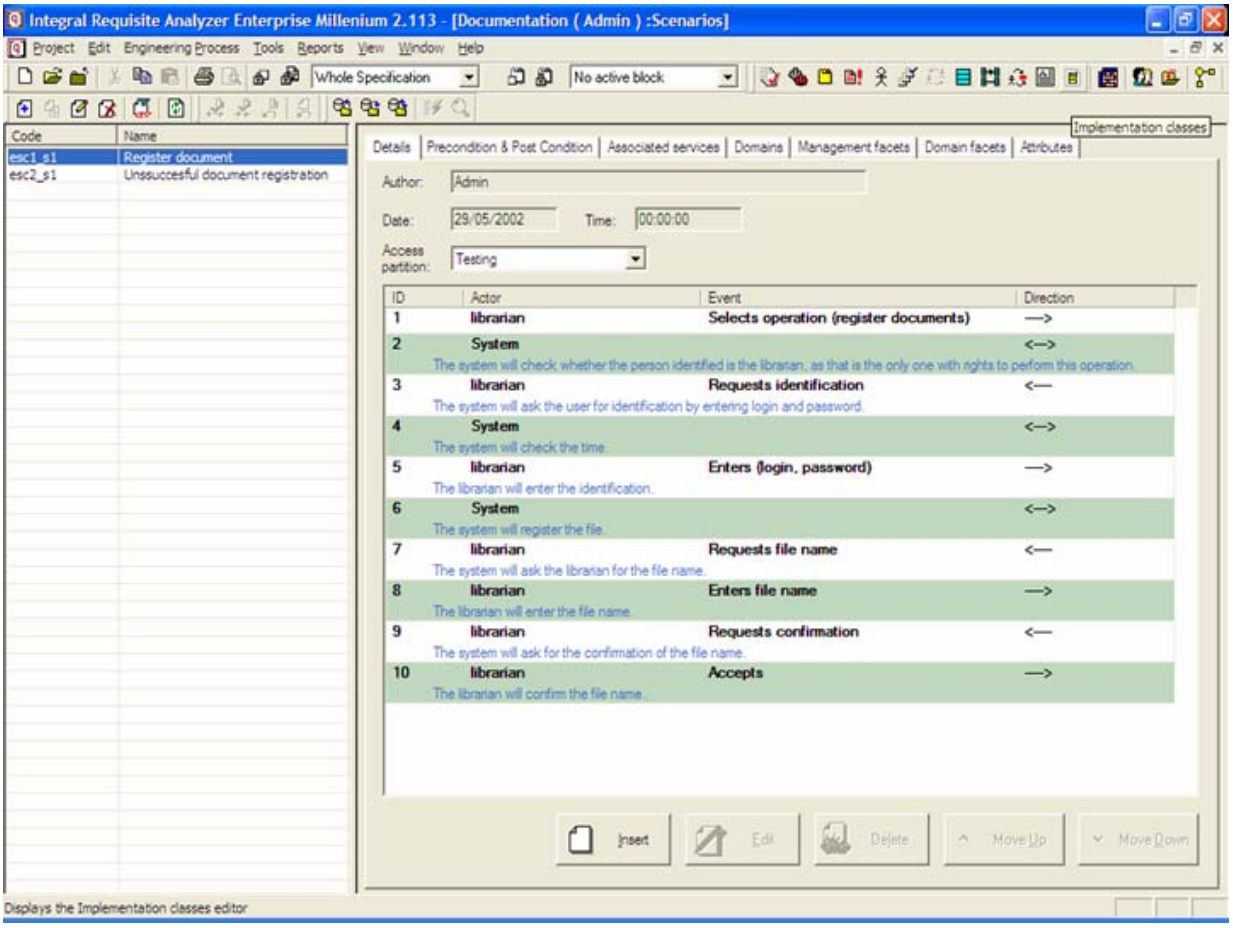


Figura 5.26 Vista de escenarios para el sistema de control de documentos.

El **caso de uso** es el principal componente de esta técnica si utiliza el enfoque orientado a objeto al especificar un sistema. La vista de los casos de uso en la herramienta IRqA proporciona un modelo de caso de uso del sistema que se esta desarrollando. Sin embargo, los casos de uso son conocidos como los servicios que el sistema proporciona y deben estar definidos antes de comenzar a realizar el diagrama de casos de uso. IRqA proporciona botones o iconos de rápido acceso para poder agregar o eliminar un actor, crear una relación, agregar o eliminar un servicio, también adiciona algunos iconos de configuración para la paleta de colores y los diagramas. En la figura 5.27 se visualiza un modelo de casos de uso para el sistema de control de documentos que se ha tomado de ejemplo.

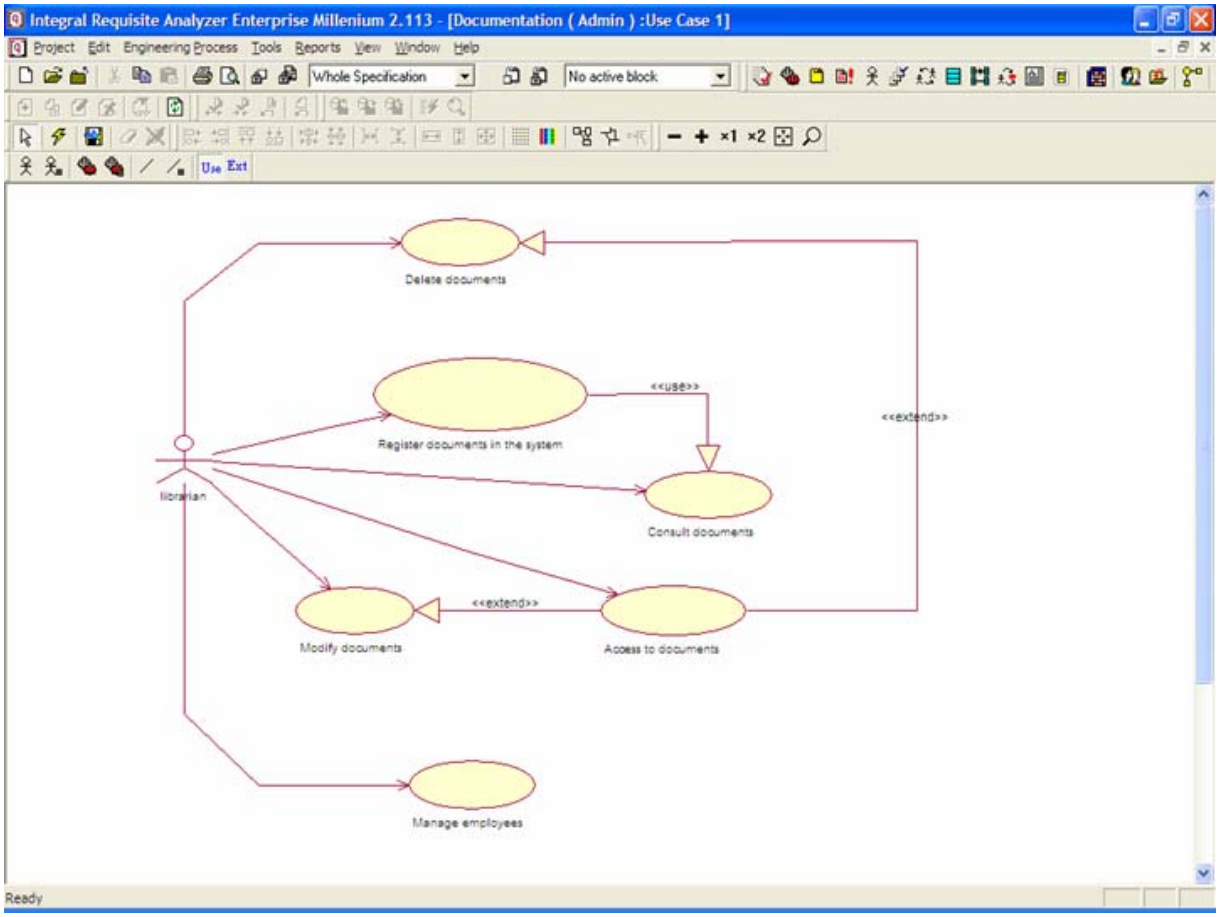


Figura 5.27 Vista del modelo de Casos de Uso.

Al igual que RequisitePro esta herramienta proporciona un enlace con *Microsoft Word* para generar plantillas de documentos de **especificación de requerimientos** del sistema. El esquema de la plantilla esta basado en el estándar de la IEEE 830-1993. Los requerimientos funcionales y no funcionales se describen mediante un código identificador que es asignado por la herramienta, el nombre del requerimiento y su descripción. En la figura 5.28 se muestra una pantalla de captura de la plantilla en Microsoft Word para la especificación de los requerimientos del ejemplo anterior.

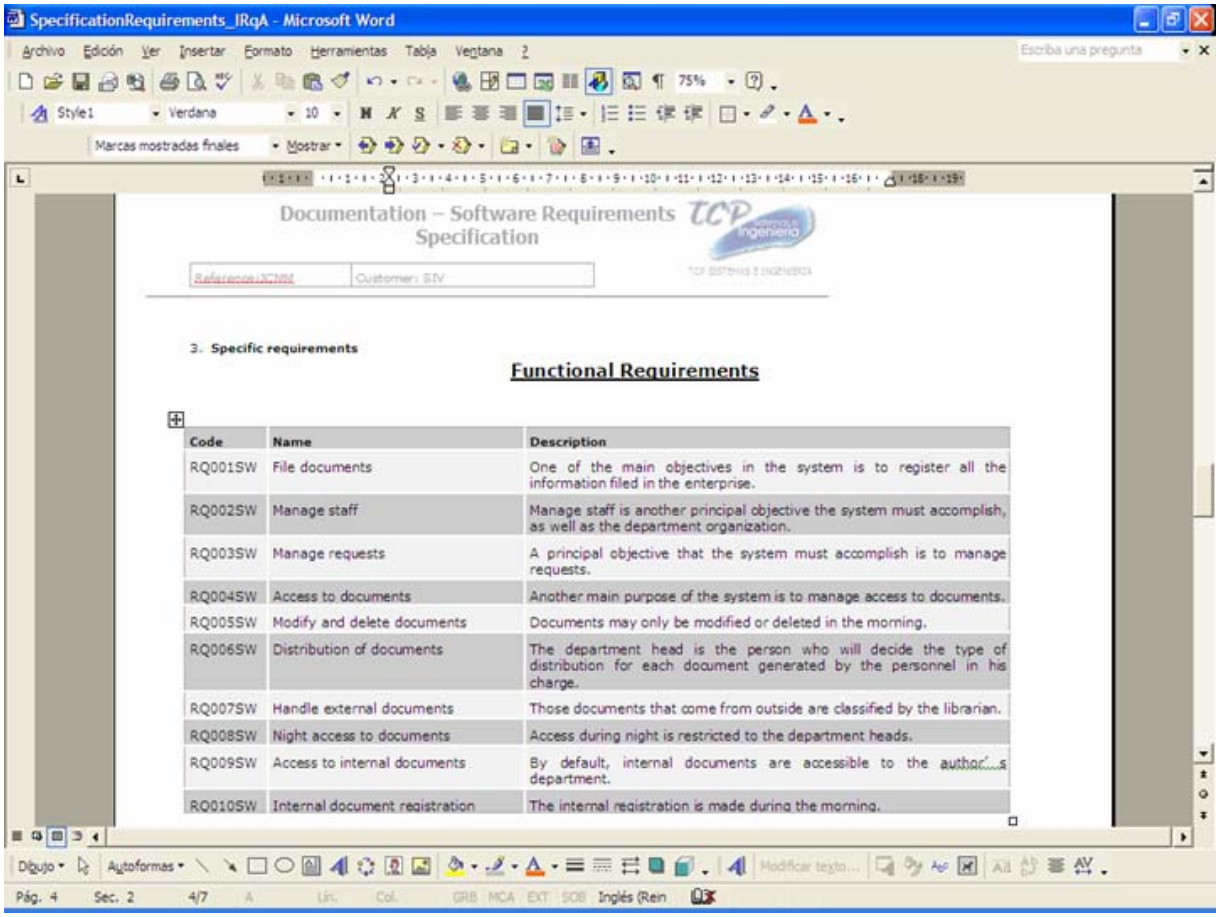


Figura 5.28 Plantilla en Microsoft Word para especificar el documento de requerimientos.

La herramienta IRqA también proporciona un mecanismo para generar **la matriz de dependencias** o rastreabilidad. Las matrices se pueden generar configurando los elementos que aparecerán en los renglones y seleccionando los elementos que estarán en las columnas, además permite la flexibilidad de elegir las etiquetas que deben incluirse en la matriz. Las matrices pueden ser generadas en la herramienta IRqA o también mediante un enlace utilizando un icono que permite abrir una hoja de cálculo en Microsoft Excel. Otra de las flexibilidades que se permiten es de incorporar un filtro para seleccionar aquellos elementos en la matriz que cumplan con restricciones impuestas, de esta manera sólo podrán mostrarse aquellos elementos que cumplan con los filtros especificados. En la figura 5.29 se muestra una matriz de dependencias entre requerimientos del sistema que se ha tomado de ejemplo. En este caso no se definen ningún filtro.

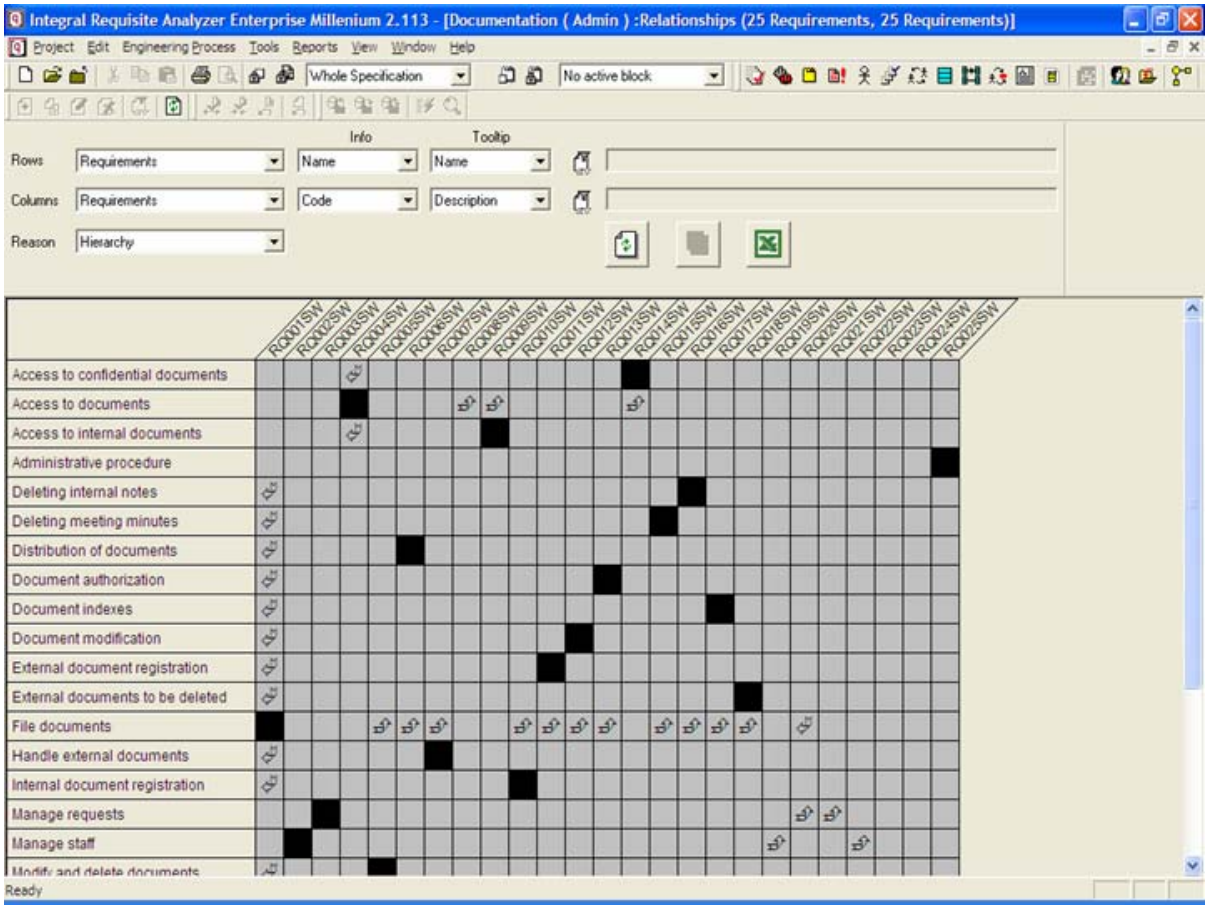


Figura 5.29 Matriz de dependencia entre requerimientos.

La seguridad e integración de los cambios en los requerimientos y de los elementos que constituyen el proyecto en IRqA se realiza mediante los **derechos de accesos** permitidos a los usuarios del proyecto. Los usuarios se encuentran organizados en *grupos* los cuales son generados por el administrador. De esta forma se pueden asignar derechos de acceso o restricciones en los diferentes elementos del proyecto. En cada grupo pueden agregarse o eliminarse *usuarios*. Cada *usuario* es definido por su nombre, teléfono, e-mail, su login y el grupo que pertenece. En la figura 5.30 se ilustra una pantalla de configuración de grupos y usuarios para el acceso a un proyecto de especificación de requerimientos en IRqA.

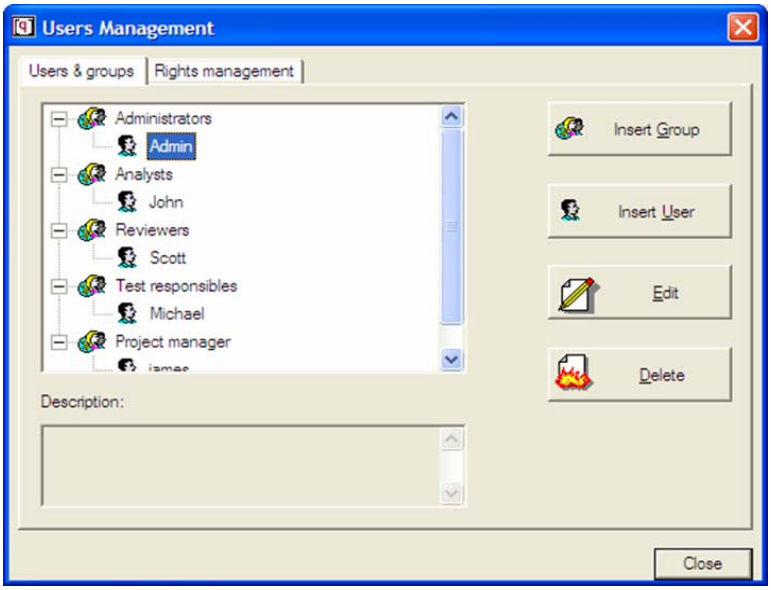


Figura 5.30 Pantalla de configuración de grupos de usuarios.

IRqA también puede generar **reportes** en línea, es decir, sin tener que salir a *Microsoft Word*. Los reportes pueden ser de dos tipos: **reportes predefinidos** o **reportes avanzados**. Los reportes predefinidos son plantillas que se encuentra integradas a la herramienta y con sólo un clic en la selección del menú de reportes se pueden generar cualquier reporte que se encuentre en la lista definida. Para los reportes avanzados permite a los usuarios de la herramienta configurar el tipo de reporte que desee, agregando detalles o filtros de restricciones. En la figura 5.31 se muestra un reporte generado a partir de los requerimientos y los servicios proporcionados por el sistema de control de documentos el cual se tomo como ejemplo para explicar el uso y las características de esta herramienta.

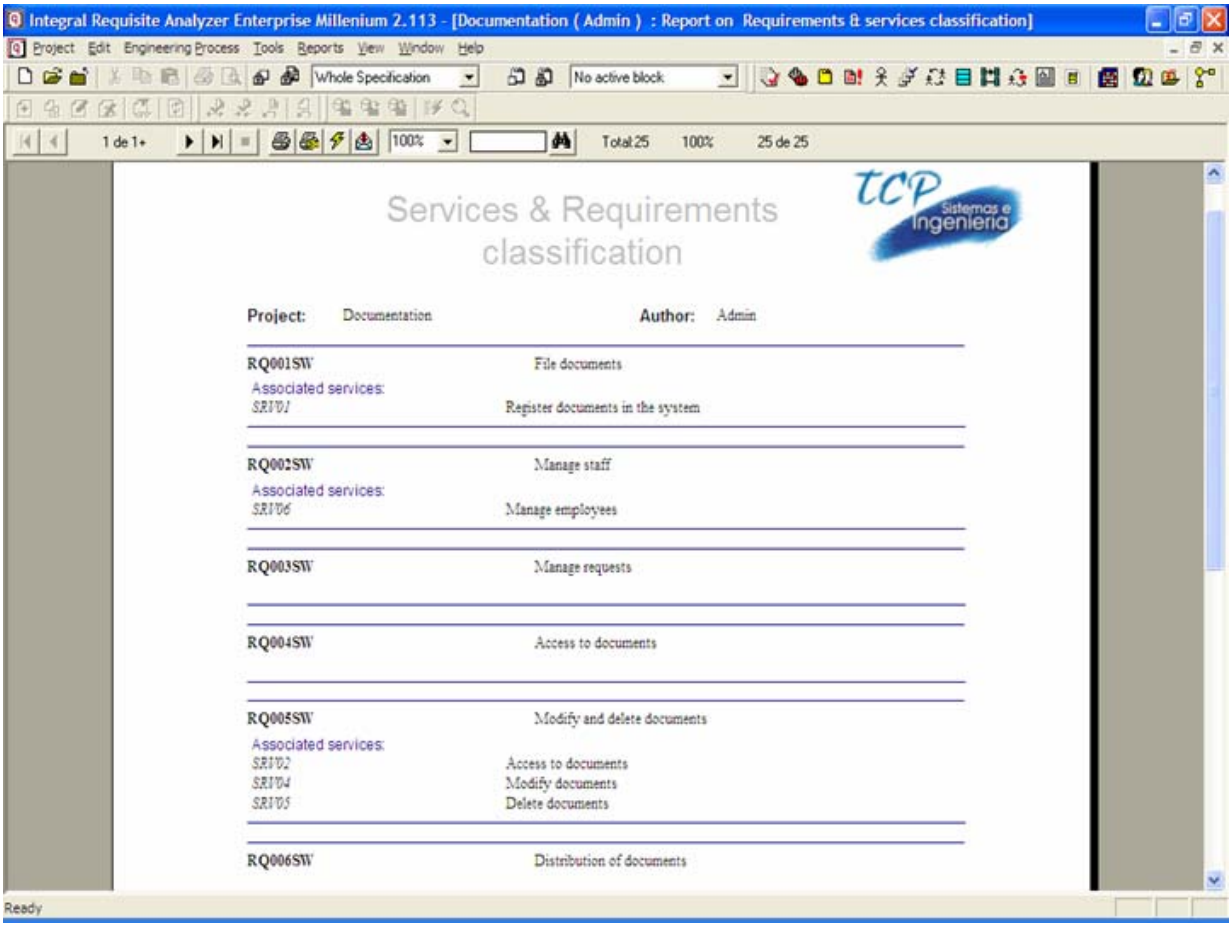


Figura 5.31 Reporte de requerimientos y servicios del sistema de control de documentos.

Capítulo 6

Análisis comparativo de Técnicas, Metodologías y Herramientas en el Proceso de Ingeniería de Requerimientos

El proceso de la Ingeniería de Requerimientos requiere de técnicas, metodologías y herramientas que faciliten a los desarrolladores realizar una adecuada administración de los requerimientos del sistema en desarrollo. En este contexto, se han presentado en capítulos anteriores una revisión de algunas técnicas, metodologías y herramientas para diferentes propósitos en la Ingeniería de Requerimientos. En este capítulo, se presenta un estudio de las diferentes técnicas que se describieron en el capitulo tres, se analizan su orientación y resaltan las ventajas de cada una de ellas en su aplicación a un caso de estudio.

6.1 Análisis comparativo de las técnicas de Obtención de Requerimientos

La obtención de los Requerimientos es una tarea compleja que requiere de un gran esfuerzo por parte de los desarrolladores para lograr comprender el sistema y las necesidades de los usuarios. Esto requiere la participación de los usuarios del sistema para lograr identificar el mayor número de requerimientos del sistema y aproximarse a una descripción completa del sistema. La tarea clave para lograr una completa especificación de requerimientos, es la obtención de los requerimientos, por lo que se deben seleccionar adecuadamente las técnicas apropiadas para este fin.

La tabla 6.1 muestra la clasificación de las técnicas para la obtención de requerimientos. La clasificación se realizó tomando en cuenta la participación de los usuarios/clientes del sistema en la obtención de los requerimientos, así como, el número de usuarios que participan y las veces que la técnica puede ser aplicada en la actividad de obtención de los requerimientos. Las clasificaciones son las siguientes.

- ***Activa.*** Los usuarios/clientes participan activamente en la obtención de los requerimientos.
- ***Pasiva.*** Los usuarios/clientes no participan o son observados por los desarrolladores.
- ***General.*** Todos los usuarios/clientes participan en la obtención de los requerimientos.
- ***Parcial.*** Solo algunos usuarios participan en la obtención de requerimientos.
- ***Iterativas.*** La técnica puede ser aplicada varias veces para obtener los requerimientos.
- ***Definitiva.*** La técnica es aplicada sólo una vez.

Técnica	Activa	Pasiva	General	Parcial	Iterativa	Definitiva
1. Entrevista	X			X	X	
2. Cuestionario	X			X		X
3. Punto de vista	X		X			X
4. Observación		X		X	X	
5. Escenarios	X		X		X	
6. Análisis de protocolos	X			X		X
7. Maestro/Aprendiz	X			X	X	
8. QFD				X		X
9. Run Use Case Workshop	X		X			X
10. Análisis de Interesados	X			X		X
11. Demostración de tareas	X			X		X
12. Enfoque grupal	X		X		X	
13. Talleres futuros	X		X		X	
14.Estudio de documentos/datos		X			X	X

Tabla 6.1 Clasificación de la técnicas de obtención de requerimientos.

Otra clasificación que se puede realizar es seleccionando las técnicas de acuerdo a su aplicación en las tareas que se desarrollan en la actividad de obtención de requerimientos. La tabla 6.2 muestra esta clasificación. Por ejemplo, para tener conocimiento del problema y entender el contexto del sistema a desarrollar se pueden aplicar la observación, entrevistas, estudio de documentos o la demostración de tareas. Con un conocimiento de la situación problemática a resolver se continua con la obtención de información que definan las características del sistema mediante la entrevista, cuestionario, análisis de protocolos y el estudio de documentos, estas técnicas se pueden emplear para recolectar suficiente información para lograr una definición inicial de las necesidades del cliente. A partir de la información obtenida se puede definir los límites del sistema mediante la técnica de escenarios, o utilizando técnicas dirigidas a un grupo de usuarios mediante reuniones con el fin de analizar el sistema y definir su alcance mediante las técnicas de Run Use Case Workshop, enfoque grupal y talleres futuros. La identificación de usuarios puede llevarse a cabo mediante las técnicas de puntos de vista, maestro/aprendiz, análisis de interesados y demostración de tareas, todas estas técnicas tienen la característica que se aplican a usuarios de manera individual. Obtenida la información suficiente se pueden definir los requerimientos del sistema mediante los puntos de vista, escenarios, maestro-aprendiz, QFD, Run Use Case Workshop, enfoque grupal y talleres futuros.

Técnica	Comprender el problema	Recolectar información	Definir límites	Identificar usuarios	Recolectar requerimientos
1. Entrevista	X	X			
2. Cuestionario		X			
3. Punto de vista				X	X
4. Observación	X	X			
5. Escenarios			X		X
6. Análisis de protocolos		X			
7. Maestro/Aprendiz				X	X
8. QFD					X
9. Run Use Case Workshop			X		X
10. Análisis de Interesados				X	
11. Demostración de tareas	X			X	
12. Enfoque grupal			X		X
13. Talleres futuros			X		X
14.Estudio de documentos – datos	X	X			

Tabla 6.2 Clasificación de la técnicas en las actividades de la obtención de requerimientos.

6.1.1 Cuadro comparativo de las Técnicas de Obtención de Requerimientos

Básicamente la comparación realizada entre las técnicas de obtención de los requerimientos es realizada de acuerdo a las ventajas y desventajas que cada técnica ofrece en la aplicación de sistemas de software interactivos, es decir, en sistemas donde la participación del usuario es necesaria para el funcionamiento del sistema. La mayoría de las técnicas requieren la participación de los usuarios para lograr obtener la mayor parte de los requerimientos del sistema.

Técnica	Ventajas	Desventajas
Entrevista	• Es utilizada como punto de partida en la obtención de los requerimientos. • Se logra conocer la opinión del cliente y los usuarios. • Se conoce la organización de la empresa. • Obtienen las metas del sistema. • Se logra un entendimiento general del problema. • Puede combinarse con otras técnicas.	• Creen perder el poder si revelan lo que saben. • No existe confianza entre los usuarios y los desarrolladores. • Utilizan diferente terminología. • Difícil de expresar. • Sabotear la entrevista por falta de cooperación. • Expresar satisfacción de la forma en que se hacen las cosas y no desea cambios. • Da mucha información irrelevante.
Cuestionario	• Se obtiene una gran cantidad de información a través del usuario. • Son flexibles. • Permite obtener datos que pueden medir resultados de forma escalar. • Permiten combinarse con otras técnicas.	• Se obtiene un gran volumen de información que requiere de análisis. • Se obtiene información redundante y en ocasiones incompletas. • Se requiere de experiencia en el dominio para tratar de comprender el comportamiento de los encuestados.
Punto de vista	• Permite estimular la aportación de ideas. • Se crea un ambiente de confianza. • Puede combinarse con otras técnicas.	• Existen personas que no les gusta hablar en público y que posiblemente tengan buenas ideas. • Se requiere de un buen facilitador que asegure el éxito de la técnica. • Puede haber requerimientos externos no encontrados.
Observación	• Contacto directo con el ambiente. • Se detectan problemas. • Puede combinarse con otras técnicas como el prototipado y los Casos de usos.	• Tiempo prolongado y difícil de permanecer en calidad de observador. • Los usuarios presentan incomodidad al ser observados.

Continuación de la tabla 6.3

Técnica	Ventajas	Desventajas
Escenarios	• Fácil de comprender y criticar un escenario. • Describe el estado inicial del escenario. • Describe el flujo de eventos. • Descubre excepciones y soluciones. • Descubre como llevar a cabo situaciones paralelas. • Describe el estado final del escenario.	• Toman tiempo de acuerdo a la complejidad del sistema. • Las personas que modelen el sistema deben conocerlo previamente.
Análisis de protocolos	• Los usuarios prefieren describir con sus propias palabras las tareas que realizan. • Se obtiene una amplia variedad de objetivos que son visualizados por los usuarios. • Permite establecer un ambiente de confianza.	• El lenguaje natural es muy ambiguo y depende mucho del contexto del dominio del sistema. • Pueden existir diferentes interpretaciones por parte del desarrollador debido al lenguaje que utiliza el usuario. • El usuario puede dejar detalles sin expresar ó que haya olvidado.
Maestro/aprendiz	• Se puede aplicar en proyectos no estructurados, para obtener el conocimiento que se encuentra en la "cabeza" del usuario. • Se puede observar condiciones que alteren el flujo normal del sistema.	• Requiere demasiado tiempo para su interpretación. • Algunos usuarios presentan incomodidad al cuestionárseles de sus tareas. • Creen que pueden perder su trabajo si revelan los detalles.
QFD	• Se concentra en maximizar la satisfacción del cliente. • Se muestra una relación directa de los requerimientos con cada caso de uso que permite su verificación en el diseño. • Pueden detectarse fácilmente problemas de inconsistencias entre requerimientos o casos de uso.	• Requiere demasiado tiempo en sistemas complejos. • Es difícil de implementar en sistemas complejos. • Se debe contar con conocimiento del dominio del sistema para su realización.
Run Use Case Workshop	• Permite la participación directa de los usuarios para definir escenarios. • Permite crear un ambiente adecuado para la participación de los demás usuarios.	• Existe el riesgo de caer en grandes diferencias de interpretación o contradicciones. • Los talleres pueden caer en ser pesados a los usuarios y generar desaliento y poca participación. • Se pueden deducir requerimientos con diferente percepción.

Continuación de la tabla 6.3

Técnica	Ventajas	Desventajas
Análisis de Stakeholders	• Se conoce la jerarquía organizacional del dominio. • Se obtienen un gran número de requerimientos desde diferentes puntos de vistas. • Se logra la participación de diferentes sectores del negocio.	• Diferentes percepciones de los requerimientos requiere de secciones de negociación y homogenización. • Se debe establecer una agenda acordada que no afecte a ningún participante. • Requiere de tiempo si la empresa es grande organizacionalmente.
Demostración de tareas	• Al usuario le es más fácil demostrar como hacen sus tareas. • Se logra una mayor interacción con el usuario. • Se pueden encontrar detalles omitidos u olvidados por los usuarios en otras técnicas. • Se logra un mayor entendimiento del sistema.	• Se requiere de la participación voluntaria del usuario. • El usuario se incomoda cuando se le cuestiona sobre su trabajo. • Requiere demasiado tiempo. • El usuario trata de hacer lo mejor posible su tarea y omite posibles defectos.
Enfoque grupal	• Se detectan los sentimientos de los usuarios hacia el sistema. • Provee un medio para hacer surgir ideas de los demás participantes. • Se logra obtener los límites y restricciones del sistema.	• Se requiere de un buen moderador para que la reunión tenga éxito. • Puede tomar tiempo si no se dirige la reunión hacia objetivos específicos. • Pueden encontrarse grupos políticos de usuarios que afecten sus intereses personales, más que a los intereses de la empresa.
Talleres futuros	• Provee un medio para hacer surgir ideas de los demás participantes. • Se logra obtener los límites y restricciones del sistema. • Permite visualizar el estado final del nuevo sistema y prever las afectaciones o restricciones que tendrá.	• Se puede exagerar en la visualización de las funciones que tendrá el futuro sistema. • Existe el riesgo de que los usuarios tengan otra percepción del sistema a como lo delimitan los desarrolladores.
Análisis de documentos/datos	• Permite realizar un planteamiento del sistema en base de formas, archivos y datos que son utilizados en el sistema actual. • Requiere solo de la participación del desarrollador sin afectar a los usuarios en sus tareas.	• No hay participación activa del usuario. • Se corre el riesgo por suposiciones de los desarrolladores en el análisis de documentos y formas.

Tabla 6.3 Cuadro comparativo de las técnicas de obtención de requerimientos.

La tabla 6.3 muestra las ventajas y desventajas que cada técnica ofrece en la actividad de obtención de los requerimientos del sistema, de esta forma permite seleccionar de manera rápida la técnica o técnicas a aplicar en el desarrollo de software, considerando el contexto del sistema y las características de los usuarios involucrados en el desarrollo.

Analizando las características de cada técnica podemos agruparlas de acuerdo a la similitud en su aplicación y en su estructura interna en seis grupos diferentes y dos más que resultan de la combinación de otros.

1. **Grupo 1**. Puede ser utilizada para cualquier enfoque ya sea funcional u orientado a objeto, la participación de los usuarios no es directamente, los desarrolladores se encargan de observar las tareas que realizan los usuarios.
2. **Grupo 2**. Una técnica clásica de obtención de requerimiento y adecuada en todo inicio de un proyecto de software, en la que el desarrollador entabla directamente una comunicación verbal con los interesados del sistema para recolectar información que le ayudará a definir sus necesidades. También suele aplicarse en las etapas restantes del proceso de desarrollo, es decir, el análisis, la validación y especificación de los requerimientos.
3. **Grupo 3**. Este grupo de técnicas, se caracteriza por la aplicación a un grupo de usuarios y el éxito depende de la habilidad del facilitador o moderador de la reunión.
4. **Grupo 4**. Este grupo de técnicas se utiliza para sistemas que son desarrollados bajo el enfoque orientado a objetos, principalmente por la aplicación de los casos de uso que son la parte esencial de este enfoque, para su aplicación es necesario contar con información recabada sobre la operación del sistema.
5. **Grupo 5**. Esta técnica se basa en la obtención de información del sistema mediante la aplicación de cuestionarios, suele combinarse con otras técnicas para lograr resultados importantes. Es útil aplicarla al inicio de todo proceso de desarrollo para obtener un contexto del sistema.
6. **Grupo 6**. La participación de los usuarios en este grupo es nula, los desarrolladores son los encargados para lograr obtener un conocimiento del sistema. Esta técnica debe ser utilizada como auxiliar de otras para obtener un mayor conocimiento del contexto del sistema, sobre todo si ya existe un sistema previo y se requiere una actualización o mejora.

7. **Grupo 7**. Este grupo combina las características del grupo uno y dos. Utiliza la observación y mediante preguntas se cuestiona al usuario para aclarar dudas de la realización de sus tareas y de esta manera lograr el conocimiento del contexto del sistema.
8. **Grupo 8**. Es el resultado de la combinación de características del grupo dos y tres, suele funcionar de una entrevista dirigida por el desarrollador hacia un grupo de usuarios o de manera particular, en busca de información o datos que le permitan definir el sistema.

Todos los grupos pueden combinarse para lograr una amplia captación de información y datos que ayuden a comprender el funcionamiento del sistema y lograr una definición de sus límites. Es difícil definir que técnica utilizar y es aún más difícil recomendar las técnicas que se deben seguir para el desarrollo de los sistemas, debido a que estos siempre tendrán características diferentes aunque se encuentre dentro de un mismo contexto. Por lo que se recomienda usar una combinación de los grupos mencionados anteriormente pero tomando en consideración la disponibilidad de los usuarios y el ambiente en donde operará el sistema que se va a desarrollar, además, si el sistema será nuevo o una mejora de uno ya existente. En la figura 6.1 de muestran los agrupamientos de las técnicas de obtención de requerimientos.

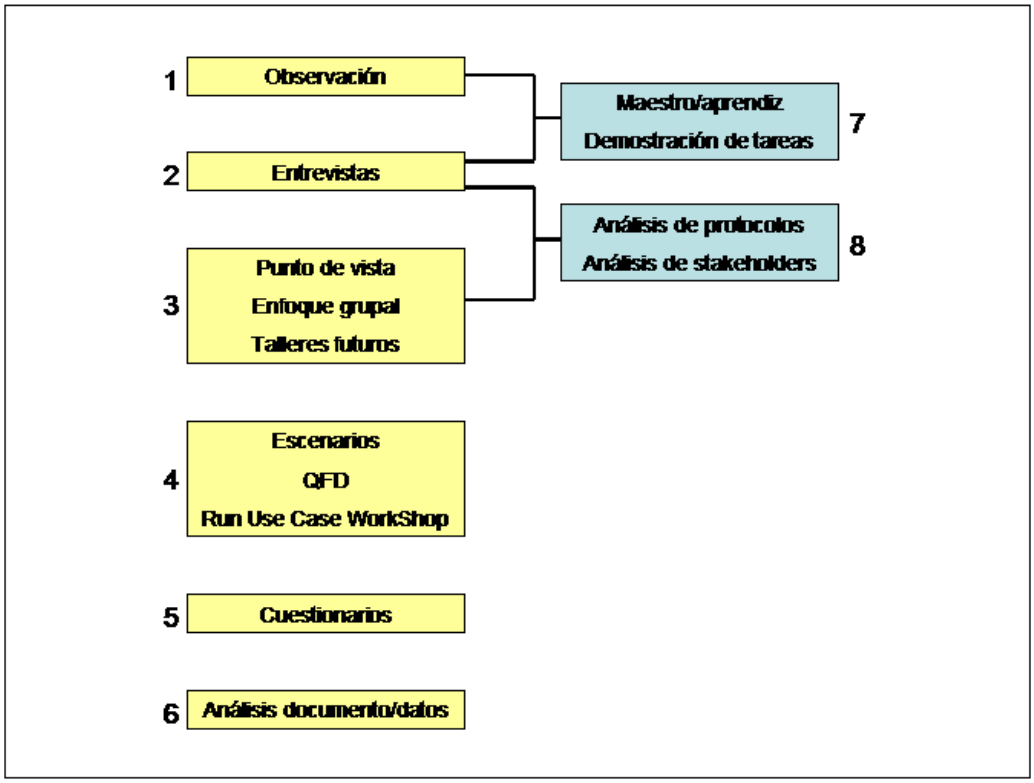


Figura 6.1 Agrupamiento de técnicas de obtención de requerimientos.

6.2 Análisis de las Metodologías para proceso de Ingeniería de Requerimientos

Las metodologías tratan de ser una guía en el desarrollo completo de un proceso, en este caso, para el proceso de Ingeniería de Requerimientos. Sin embargo, no todas logran comprender las cuatro actividades que se han definido en esta investigación para el proceso de Ingeniería de Requerimientos. Otra característica que presentan es la orientación hacia un enfoque de desarrollo y la notación que utilizan. Por estas y otras consideraciones, se ha desarrollado un estudio de estas herramientas del cual se obtuvo la tabla 6.4, donde se abrevian las características más importantes de cada herramienta con el objetivo de facilitar una selección adecuada de la metodología que se ha de aplicar en la obtención del documento de especificación de requerimientos.

Metodología	**Notación**	**Actividades**				**Orientación**
		O	**A**	**E**	**V**	
Joint Application Design	NG	x	-	x	x	-
Cooperative Requirements Capture	NG	x	x	x	-	-
Objectory	NG	-	x	x	x	O
COHERENCE	LN, NG	-	x	x	-	O-Pv
Soft System Methodology	NG	x	x	x	-	Pv
Análisis Estructurado	NG	x	x	x	-	F
Metodología de DSED	NG	-	x	x	-	F/O
COLOR-X	LN, NG	-	x	x	x	O
RARE-IDIOM	LN, NG	x	x	x	x	R
El método SADT	NG	-	x	x	-	F-Pv
El método CORE	NG	-	x	x	-	F-Pv
El método VOSE	LN, NG	-	x	x	-	Pv
El método VORD	NG	x	x	x	-	F-Pv

Tabla 6.4 Análisis de las metodologías para el proceso de Ingeniería de Requerimientos.

Actividades: **O** = obtención, **A** = análisis, **E** = especificación y **V** = validación.

Notación: **LN** = Lenguaje Natural y **NG** = Notación Gráfica.

Orientación de la metodología: **F** = funcional, **O** = objeto, **R** = reuso y **Pv** = puntos de vista.

Cada metodología cumple con mostrar las directrices para realizar sistemáticamente el proceso de Ingeniería de Requerimientos, sin embargo no todas cubren todas las actividades del proceso, como se ilustra en la tabla 6.4. Únicamente la metodología RARE-IDIOM comprende las cuatro actividades pero su orientación es hacia el desarrollo utilizando el reuso de componentes. Es evidente que la mayoría de las metodologías utilizan una notación gráfica para la descripción del funcionamiento del sistema. El uso de la notación gráfica parece ser más apropiado que la notación en lenguaje natural, debido a que ofrece una visión atractiva y hace comprensible el documento de especificación de requerimiento para cualquier persona. De esta manera se logra reducir las ambigüedades generadas por el lenguaje natural en la especificación de los requerimientos. Se pueden utilizar diferentes modelos gráficos para complementar en detalles el funcionamiento del sistema.

Por otro lado, se considera que la metodología es completa si comprende las cuatro actividades definidas en el proceso de Ingeniería de Requerimientos, la orientación a un enfoque específico juega un papel decisivo en la selección de una metodología, ya que depende de la experiencia de los desarrolladores y el enfoque que deseen darle al producto final. Sin embargo, hay metodologías que permiten la flexibilidad de elegir entre dos enfoques diferentes, otras están definidas mediante el uso de un solo enfoque y definen las técnicas de captura de requerimientos. Por ejemplo los métodos SADT, CORE, VOSE y VORD que definen un enfoque funcional mediante el uso de la técnica de puntos de vista.

6.3 Análisis de las Herramientas de software para el proceso de Ingeniería de Requerimientos

Aunque algunas metodologías definen su herramienta a utilizar, se evaluaron cuatro herramientas de software para la administración de los requerimientos. Para realizar la evaluación se establecieron criterios para medir su rendimiento y desempeño [1]. Los criterios considerados son los siguientes.

1. **Aplicabilidad** ¿Describe problemas y soluciones del mundo real de manera natural y realista? ¿Es compatible con otra que se utilizan en el desarrollo del sistema?
2. **Capacidad de implementación** ¿Puede traducirse la especificación a una implementación? ¿Es automatizada? ¿Se puede generar código? ¿Es código eficiente?
3. **Capacidad de comprobación y simulación** ¿Puede utilizarse la especificación para probar la implementación? ¿Todos los requerimientos son verificables por la implementación?
4. **Capacidad de contraste** ¿Puede un usuario contrastar la exactitud de la especificación? ¿La lectura de la especificación es accesible para todos los usuarios?
5. **Facilidad de mantenimiento** ¿Es fácil de cambiar la especificación a medida de que el sistema evolucione? ¿Será útil la especificación para las actividades del mantenimiento?
6. **Nivel de abstracción y capacidad de expresión** ¿La especificación describe expresivamente los objetos reales? ¿La especificación describe expresivamente el dominio?
7. **Profundidad** ¿Se pueden detecta inconsistencia? ¿Se pueden detectar ambigüedades?
8. **Madurez de la herramienta** ¿La herramienta es de alta calidad? ¿Existen manuales o ayuda en línea para aprenderlas?
9. **Curva del aprendizaje** ¿Puede un usuario novato aprender rápidamente los conceptos, la sintaxis y las semánticas?
10. **Disciplina** ¿Obliga a escribir especificaciones bien estructuradas, comprensibles y bien formalizadas?

6.3.1 Evaluación de las Herramientas de Administración de los Requerimientos de Software

Los criterios que se definieron anteriormente se utilizaron para llevar a cabo una evaluación de las herramientas de software que fueron definidas en el capítulo cuatro. La tabla 6.4 corresponde a la evaluación de las herramientas utilizando los criterios antes definidos, la calificación corresponde un 1 si la herramienta cumple con esta característica, y un 0 si no cumple con el criterio de evaluación.

Criterios	V	R	CE	I
¿Describe problemas y soluciones del mundo real de manera natural y realista?	1	1	1	1
¿Todos los requerimientos son verificables por la implementación?	1	1	0	1
¿Puede traducirse la especificación a una implementación?	0	1	0	0
¿Es automatizada?	1	1	1	1
¿Se puede generar código?	0	1	0	0
¿Es código eficiente?	0	1	0	0
¿Puede utilizarse la especificación para probar la implementación?	0	1	1	1
¿Todos los requerimientos son verificables por la implementación?	0	1	1	1
¿Puede un usuario contrastar la exactitud de la especificación?	1	1	1	1
¿La lectura de la especificación es accesible para todos los usuarios?	1	1	1	1
¿Es fácil de cambiar la especificación a medida de que el sistema evolucione?	0	1	1	0
¿Será útil la especificación para las actividades del mantenimiento?	1	1	1	1
¿La especificación describe expresivamente los objetos reales?	1	1	1	1
¿La especificación describe expresivamente el dominio?	1	1	1	1
¿Se pueden detecta inconsistencia?	1	1	0	1
¿Se pueden detectar ambigüedades?	0	0	0	0
¿La herramienta es de alta calidad?	0	1	1	1
¿Existen manuales o ayuda en línea para aprenderlas?	1	1	1	1
¿Puede un usuario novato aprender rápidamente los conceptos, la sintaxis y las semánticas?	0	0	0	0
¿Obliga a escribir especificaciones bien estructuradas, comprensibles y bien formalizadas?	1	1	1	1
Totales	**11**	**18**	**13**	**14**

Tabla 6.5 Evaluación de las herramientas.

En la tabla 6.5 se muestran las columnas etiquetadas con las letras **V**, **R**, **CE** e **I**, corresponden a las herramientas VORDTools, RequisitePro, Catalize Enterprice e IRqA respectivamente. Considerando los totales se puede decidir que herramienta es la más completa en base a los criterios evaluados.

Capítulo 7

Caso de Estudio: El Documento de Especificación de Requerimientos del SIV

El caso de estudio que se desarrolló, para la Especificación de los Requerimientos utilizando las técnicas y herramientas revisadas en capítulos anteriores, es el Sistema de Inscripción Virtual (SIV), que permitirá a los alumnos del Departamento de Ingeniería Eléctrica (DIE) del CINVESTAV-IPN inscribirse a los cursos de manera remota mediante el uso de Internet. Se obtuvo el Documento de Especificación de los Requerimientos del SIV, donde se encuentran expresadas las características y las restricciones del sistema, mediante la aplicación de las actividades concernientes al proceso de Ingeniería de Requerimientos definidas en el capítulo tres de esta investigación.

7.1 Introducción

En este documento se pretende definir de manera clara y precisa los servicios y restricciones que proveerá el Sistema de Inscripción Virtual (SIV), que permitirá a los alumnos del Departamento de Ingeniería Eléctrica (DIE) del CINVESTAV-IPN inscribirse a los cursos de manera remota mediante el uso de Internet.

7.1.1 Propósito del Documento de Especificación de Requerimientos del SIV

El Documento de Especificación de Requerimientos (DER) del SIV, tiene como finalidad de presentar una descripción homogénea de los requerimientos del sistema aquí definidos, considerando que los lectores posean o no conocimientos técnicos para su comprensión. Además, que pueda servir como un contrato entre "*el cliente"* y "*el desarrollador"*, debido a que se encuentran definidas las características del sistema que se va a desarrollar.

La Ingeniería de Requerimientos como parte del proceso de desarrollo de software, es un puente importante hacia el diseño, por lo que la correcta descripción de las necesidades de los usuarios en el DER define el éxito o fracaso del sistema.

7.1.2 Alcance del Documento de Especificación de Requerimientos del SIV

El DER contiene una definición de los requerimientos funcionales y no funcionales para el SIV. Se utiliza el lenguaje natural estructurado mediante formas y esquemas definidos para definir éstos requerimientos. La definición posteriormente podrá ser analizada y discutida por los diferentes tipos de usuarios involucrados en el SIV, con el objetivo de buscar conflictos entre requerimientos y resolverlos si es que se encuentran problemas. Después de haberse realizado lo anterior, se especifican los requerimientos con mayor detalle, utilizando modelos de representación para obtener diferentes perspectivas del sistema. La siguiente etapa es validar la especificación de los requerimientos utilizando criterios de validez, consistencia y ambigüedad. Finalmente hablamos sobre la evolución del sistema, los puntos que hay que cuidar y las tendencias del sistema.

7.2 Descripción General

En este apartado se describe como se desarrolla actualmente el Sistema de Inscripción a cursos que realizan los alumnos del DIE, se presenta una propuesta para el SIV, definiendo el propósito y alcance. Se identifican y definirán los posibles usuarios interesados en el SIV con el fin de trabajar conjuntamente con el equipo de desarrollo para especificar sus necesidades.

7.2.1 Descripción del sistema actual

Es necesario describir el proceso actual de inscripción de los alumnos a los cursos del departamento de Ingeniería eléctrica, como un punto de partida y referencial para obtener y especificar los requerimientos, también para tener antecedentes y poder contrastar el sistema actual con respecto al alcance del nuevo sistema (SIV).

El proceso actual de inscripción de alumnos a cursos del programa académico del departamento de Ingeniería Eléctrica se desarrolla tradicionalmente de la siguiente forma:

1. *Cada profesor tiene asignado un número de alumnos.*

Cuando los alumnos son de nuevo ingreso, se asigna a un integrante del colegio de profesores, un número de alumnos que regularmente es de tres alumnos. Esta asignación es de acuerdo a la especialidad que el alumno pretende seguir, y se lleva a cabo con el fin de que los alumnos tengan una guía y asesoría personalizada en la elección de los cursos en su curricula del programa de postgrado, es decir, una orientación general del panorama académico. El asesor es el mismo durante el tiempo que dura el programa académico. Puede haber cambios pero estos se consideran casos especiales.

2. *El coordinador académico y el Colegio de Profesores realizan la programación de cursos de acuerdo al reglamento y al programa de la maestría.*

De acuerdo a las inquietudes manifestadas por los alumnos el colegio de profesores y el coordinador programan los cursos que pueden impartir en el presente cuatrimestre. Esta programación se lleva a cabo en base de un programa académico y a un reglamento del postgrado de cada sección.

3. *Se hace una previa asignación a la curricula de los alumnos, de acuerdo al perfil universitario.*

Después de tener la curricula de los cursos que se van a impartir en todo el año el profesor realiza una asignación previa. Se toma como base la formación académica del alumno y los intereses de especialización de este. Otro punto que se revisa, es que los alumnos cubran las materias del núcleo académico del programa de Postgrado.

4. *Se integra la asignación previa al expediente del alumno.*

Toda la información de los alumnos es controlada mediante expedientes, generalmente un expediente consta de documentos de antecedentes referente a cada alumno, currículum, lista de cursos, perfil, y otros más; separados en carpetas que son almacenadas en un archivero que controla una secretaria administrativa.

5. *El alumno y el asesor revisan la curricula de los cursos para el alumno.*

Los alumnos son llamados a entrevistas personales con sus asesores. El objetivo es, programar los cursos que llevará en el transcurso del programa académico, esta programación se realiza tomando en cuenta también la previa selección, mencionada en el punto 3. El alumno recibe la orientación de su asesor con más detalle del panorama de su especialización.

6. *Se integra la curricula al expediente del alumno.*

El resultado de la entrevista es la integración del documento o formato que contiene las materias programadas y es entregado al personal administrativo para que realice las inscripciones a los cursos correspondientes. Una copia de estas inscripciones es integrada al expediente del alumno.

7. *Transcurre todo un mes en el cual se pueden realizar modificaciones por parte del alumno ó el asesor y en último caso, por el coordinador académico.*

Se recomienda que el alumno asista a 1 ó 2 cursos además de los obligatorios. Estos cursos pueden ser de acuerdo al criterio de cada alumno, con la idea de que evalué los cursos que tiene asignados. En caso de querer realizar algún cambio de curso o materia, es necesario notificarlo con el asesor, o en ausencia del asesor, con el coordinador para la autorización de dicho cambio. El tiempo en el cual se pueden realizar cambios es de un mes.

8. *La información de cada sección es entregada a Servicios Escolares del Cinvestav–IPN.*

Después de este proceso, la información de las inscripciones de todos los alumnos es entregada a Servicios Escolares del CINVESTAV-IPN, para su selección y tratamiento, y que ya no corresponde a las secciones del departamento de Ingeniería Eléctrica.

Esta información se obtuvo de la entrevista a las secretarias de las secciones correspondientes al Departamento de Ingeniería Eléctrica del CINVESTAV-IPN.

7.2.2 Propósito y alcance del SIV

El propósito del SIV es básicamente permitir a los alumnos del Departamento de Ingeniería Eléctrica, realizar inscripciones a los cursos que se imparten en dicho departamento de forma remota. El SIV, permitirá agilizar la inscripción a los cursos que los alumnos elijan de la sección a la que pertenezcan.

DEFINICIÓN DEL PROBLEMA

Se solicita al equipo de desarrollo de software especificar, diseñar e implementar un sistema de inscripciones virtual, que permita al Departamento de Ingeniería Eléctrica realizar inscripciones a cursos de forma remota. El sistema SIV, permitirá agilizar la inscripción a los cursos que imparte el Departamento de Ingeniería Eléctrica del CINVESTAV-IPN a través de Internet. Asimismo, el sistema SIV, permitirá al alumno elegir los cursos que desea tomar dentro de la sección a la que pertenezca dentro del Departamento.

La entrega del software y los documentos necesarios será realizada en las fechas establecidas en común acuerdo entre el cliente y el equipo de desarrollo.

REQUERIMIENTOS BÁSICOS

El sistema SIV debe proveer la siguiente funcionalidad:

- El sistema SIV permitirá a un alumno elegir los cursos que desea tomar en un cuatrimestre, en la sección del Departamento de Ingeniería Eléctrica a que pertenezca.
- Sólo podrá hacer uso del sistema SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares (que no estén dados de baja) en alguna sección del Departamento de Ingeniería Eléctrica (DIE).

- Todos los accesos al sistema SIV deberán realizarse desde una interfaz gráfica accesible desde Internet.
- El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materia.
- Así mismo, el sistema permitirá la creación y modificación de bases de datos conteniendo información de los cursos y de los alumnos inscritos en cada sección del DIE del CINVESTAV-IPN.
- El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.
- Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrán acceder la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.
- Se pueden considerar las siguientes vistas al sistema:
  - Público en general (los cuales solo pueden consultar datos de los cursos).
  - Alumnos del CINVESTAV-IPN (los cuales pueden consultar e inscribirse a los distintos cursos que les permite el reglamento).
  - Coordinadores Académicos quienes tienen todos los permisos para crear las bases de datos de alumnos y cursos y agregar/modificar su contenido.

BASES DE DATOS

Las bases de datos podrán ser creadas y/o modificadas por el coordinador mediante un manejador de bases de datos convencional.

La información a incluir en las bases de datos es:

- *ALUMNOS: Fecha actual, Fecha de Inscripción a la Sección, Datos Bibliográficos, Universidades o Colegios donde estuvo inscrito antes el alumno, resultado del examen de admisión a la sección, Beca del alumno, Asesor asignado.*
- *CURSO: Nombre del curso, profesor que imparte el curso, cuatrimestre en que se imparte, Contenido del curso, cursos de prerrequisito, número de alumnos registrados a este curso.*

INTERFAZ DE USUARIO

Al inicio del sistema, se leerá la información de estas dos bases de datos. La interfaz de usuario será capaz de:

- Presentar un menú basado en ventanas y botones que permita desplegar los alumnos con sus datos respectivos y desplegar los cursos y la información que corresponde a cada curso.
- Permitir a distintos tipos de usuarios (Coordinadores, alumnos y público) introducir y leer información del sistema, presentando distintas vista.
- Permitir al coordinador crear las bases de datos, ver que alumno están inscritos en cada curso, ver en que curso se inscribió y modificar e imprimir el contenido de las bases de datos.
- Permitir al coordinador académico de alguna sección asignar una clave de acceso al sistema a cada alumno.
- Permitir a los alumnos seleccionar una sección y un cuatrimestre del Departamento de Ingeniería Eléctrica en el cual desea inscribirse a llevar un conjunto de materias.

SEGURIDAD

El alumno podrá inscribirse a los cursos si se cuenta con un contraseña asignada por el coordinador académico. Sólo podrá inscribirse a un número máximo de curso por cuatrimestre (de acuerdo a lo establecido por el reglamento). Además sólo podrá inscribirse dentro de las fechas establecidas previas al inicio del cuatrimestre correspondiente.

El alumno podrá salvar el estado del SIV en cualquier momento. Si el alumno no termina de inscribirse a todas las materias que debe cursar en el cuatrimestre, podrá hacerlo en una sección futura, siempre y cuando sea antes de las fechas establecidas previas al inicio del cuatrimestre correspondiente. Además si el alumno no salva el estado de su sección al salirse, se le presentará una advertencia, avisándole que no ha salvado y permitiéndole que salve o descarte los cambios hechos.

SUGERENCIAS

Se recomienda consultar el reglamento del CINVESTAV-IPN, en lo que corresponde a inscripciones. Así mismo se debe consultar en todas las secciones del Departamento de Ingeniería Eléctrica sobre los cursos que se ofrecen en el próximo año escolar.

7.2.3 Contexto del SIV

El sistema SIV debe permitir a los alumnos realizar las inscripciones a los cursos que se imparten en el DIE del CINVESTAV-IPN de manera virtual, es decir, que cualquier alumno inscrito en el DIE pueda realizar la inscripción a los cursos mediante el acceso al SIV desde cualquier lugar usando Internet. Se pretende que el SIV proporcione un servicio de calidad y se optimicen los tiempos que requiere la inscripción de manera tradicionalmente, además de reducir la tasa de errores. En la figura 7.1 se presenta una descripción general del funcionamiento del sistema propuesto para el SIV. Se puede observar la relación general de los componentes del sistema trabajando en conjunto. Puede apreciarse el flujo de información de una manera general. El esquema de interconexión entre el usuario y servidor se realiza mediante la red Internet.

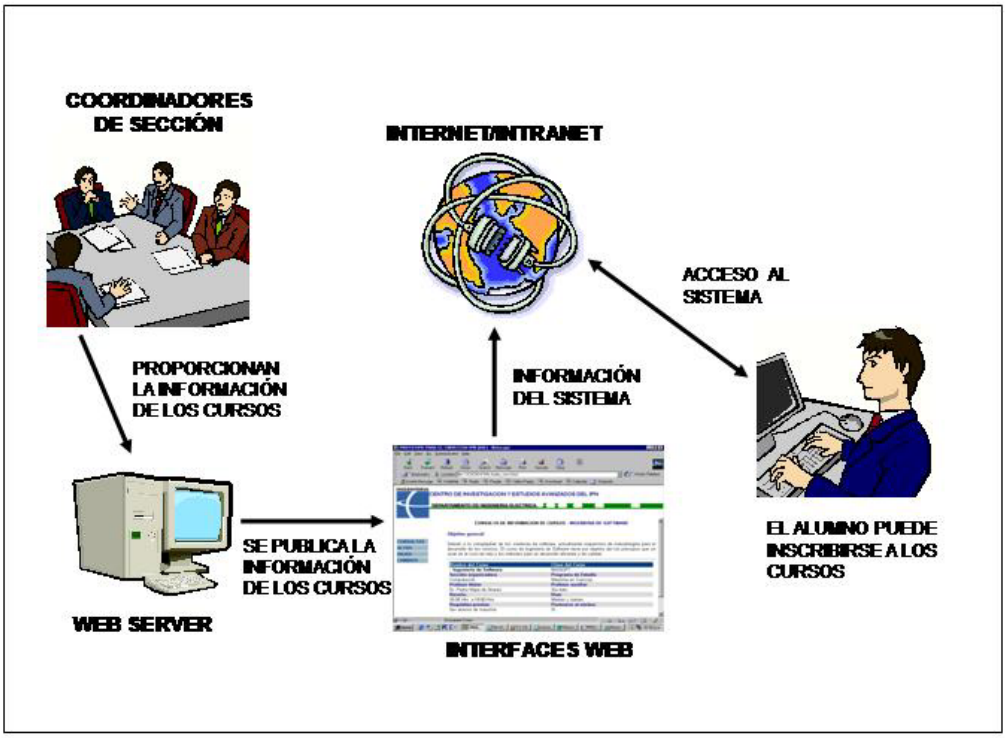


Figura 7.1 Descripción general del SIV.

7.2.4 Identificación de los usuarios involucrados (stakeholders)

Los tipos de usuarios involucrados en la especificación de los requerimientos se encuentran especificados en la tabla 7.1.

Usuario entrevistado	Cargo
Dr. Pedro Mejía Álvarez	Contratista del SIV
Dr. Jorge Buenabad Chavez	Coordinador Académico de la sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN
Sofía Reza Cruz	Secretaria encargada de Control Escolar de alumnos, en la Sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN
Juan Carlos Medina Martínez	Alumno del CINVESTAV-IPN

Tabla 7.1 Identificación de usuarios.

7.3 Definición de los Requerimientos

La definición de los requerimientos es una declaración en lenguaje natural en la que se describen las necesidades de los usuarios del SIV, estos requerimientos fueron obtenidos a partir de la definición inicial del cliente y mediante entrevistas y cuestionarios realizados a varios stakeholders del DIE (vea apéndice A).

7.3.1 Definición de Requerimientos Básicos del SIV

Los requerimientos básicos del SIV se obtuvieron de la definición inicial que proporcionó el cliente del sistema, además de algunos otros que fueron propuestos por el equipo de desarrollo considerando la información que se obtuvo de la lectura de documentos relacionados y entrevistas realizadas a los usuarios del sistema.

Id	Descripción	Prioridad	Origen
RB1	El sistema SIV permitirá a un alumno elegir los cursos que desea tomar en un cuatrimestre, en la sección del Departamento de Ingeniería Eléctrica a que pertenezca.	Establecido	Definición del sistema
RB2	Sólo podrá hacer uso del sistema SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares	Establecido	Definición del sistema
RB3	Todos los accesos al sistema SIV deberán realizarse desde una interfaz gráfica mediante menús, ventanas y botones, que sea accesible desde Internet.	Establecido	Definición del sistema
RB4	El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materia. Sólo se podrán inscribirse a un número máximo de cursos por cuatrimestre y dentro de las fechas establecidas.	Establecido	Definición del sistema
RB5	El sistema permitirá la creación y modificación de las bases de datos que contienen la información de los cursos y de los alumnos inscritos en cada sección del DIE del CINVESTAV-IPN.	Establecido	Definición del sistema
RB6	El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.	Establecido	Definición del sistema
RB7	Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrá acceder la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.	Establecido	Definición del sistema
RB8	El sistema permitirá al público en general ver los cursos que se imparten en las secciones del DIE	Establecido	Definición del sistema
RB9	Se pueden considerar las siguientes vistas al sistema: público en general, alumnos, coordinadores académicos.	Establecido	Definición del sistema

Tabla 7.2 Definición de los requerimientos básicos del SIV.

Continuación de la **tabla** 7.2

Id	Descripción	Prioridad	Origen
RB10	El sistema debe presentar una advertencia al usuario avisando que no ha salvado el estado del sistema y permitiéndole que salve o descarte los cambios hechos.	Establecido	Definición del sistema
RB11	El sistema permitirá al usuario investigador académico ver sus datos y los cursos que imparte, además de la lista de alumnos que toman sus cursos y los alumnos asignados para asesorias.	Propuesto	Documentos y entrevistas
RB12	El sistema permitirá al coordinador agregar, modificar sus datos y eliminar a un investigador de su sección.	Propuesto	Documentos y entrevistas
RB13	El coordinador podrá ver los cursos que imparte cada investigador de su sección	Propuesto	Documentos y entrevistas
RB14	El coordinador podrá ver todos los cursos que se imparten en el cuatrimestre.	Propuesto	Documentos y entrevistas
RB15	El sistema permitirá al alumno visualizar todos los cursos que imparte cada investigador de la sección	Propuesto	Documentos y entrevistas
RB16	El sistema debe permitir al alumno cambiar sus cursos que ha elegido para el cuatrimestre, verificando que los cambios sean dentro de las fechas establecidas.	Propuesto	Documentos y entrevistas
RB17	El sistema permitirá al público en general ver los cursos que imparte cada investigador de las secciones del DIE.	Propuesto	Documentos y entrevistas

La tabla 7.2 corresponde a la definición de los requerimientos básicos para el SIV, se utiliza un identificador formado por dos letras y un número para identificar un requerimiento, donde la primera letra describe a un **requerimiento** y la segunda que es **básico**, el número se debe a la secuencia de cómo se encontraron los requerimientos en la definición del problema. La prioridad en la tabla de los requerimientos básicos se refiere a que el requerimiento puede ser **establecido** por el cliente o **propuesto** por los desarrolladores. Finalmente la columna de origen describe el lugar donde se encuentra el requerimiento.

7.3.2 Definición de Requerimientos Funcionales del SIV

Los requerimientos encontrados para el SIV se obtuvieron de la depuración de los requerimientos básicos proporcionados por el cliente, de las entrevistas y cuestionarios aplicados a los distintos usuarios del SIV y de la recopilación de información en documentos y reglamentos del DIE. Los requerimientos se encuentran especificados mediante un identificador, su descripción y el origen de donde se obtuvieron para futuras referencias.

ID	**Requerimientos Funcionales: Tipos de usuario en el SIV**	**Origen**
RF1	El sistema debe validar el ingreso de los **usuarios** mediante un *login* y *contraseña,* excepto para el usuario público en general que no requiere login ni contraseña. Un usuario puede ser un alumno, un coordinador, un investigador académico o una persona del público en general.	RB9
RF1.1	El SIV debe identificar el tipo de usuario y dar servicio de acuerdo a su perfil.	
ID	**Requerimientos Funcionales: Servicios al usuario Coordinador**	**Origen**
RF2	El usuario **coordinador** podrá realizar los siguientes servicios:	
RF2.1	El usuario **coordinador** podrá asignar el login y contraseña para los usuarios **alumnos**, **coordinadores** e **investigadores académicos** de la sección a la que pertenece.	RB6
RF2.2	El coordinador podrá agregar a un alumno al sistema, esto implica registrar todos sus datos.	RB5 y RB7
RF2.3	El coordinador podrá agregar los datos de un curso de la sección en el SIV, sin registrar los detalles del curso.	RB5 y RB7
RF2.4	El coordinador podrá agregar a un investigador de su sección al sistema, esto implica registrar todos sus datos personales.	RB12
RF2.5	El coordinador podrá agregar a un coordinador al sistema, esto implica registrar todos sus datos y atributos.	E. y C.
RF2.6	El coordinador podrá modificar los datos de un alumno de su sección que se encuentre registrado en el sistema. También podrá cambiar los cursos que el alumno eligió para su cuatrimestre actual.	RB7
RF2.7	El coordinador podrá modificar la información de los curso de su sección.	RB7
RF2.8	El coordinador podrá modificar los datos de un investigador de su sección en la que se encuentre registrado en el sistema.	E. y C.

RF2.9	El coordinador podrá modificar las contraseñas de los usuarios alumno e investigador académico que pertenezcan a su sección y estén registrados en el SIV.	RB6 y RB7
RF2.10	El coordinador podrá eliminar cualquier curso de su sección.	RB7
RF2.11	El coordinador podrá eliminar a un alumno de la sección.	RB7
RF2.12	El coordinador podrá eliminar a un investigador de su sección que se encuentre registrado en el sistema.	RB12
RF2.13	El coordinador podrá establecer el periodo de inscripción a los cursos para cada cuatrimestre. El periodo válido de inscripción comprende 30 días a partir del inicio de cada cuatrimestre.	E. y C.
RF2.14	El coordinador podrá ver los cursos que cada alumno escogió para el cuatrimestre actual.	Propuesto
RF2.15	El coordinador podrá ver la lista de alumnos que se inscribieron a un curso en específico.	Propuesto
RF2.16	El coordinador podrá ver los cursos que imparte cada investigador de su sección.	RB13
ID	**Requerimientos Funcionales: Servicios al usuario Alumno**	**Origen**
RF3	El **SIV** permitirá al **alumno**, que se encuentre registrado en el departamento de Ingeniería Eléctrica del CINVESTAV-IPN y sea alumno regular, realizar los siguientes servicios:	RB1
RF3.1	El usuario alumno podrá inscribirse a los cursos registrados en el sistema para el cuatrimestre actual de su sección. El sistema comprobará que la inscripción la realice en el periodo establecido, además de comprobar que el alumno seleccione un máximo de 4 cursos por cuatrimestre.	RB4
RF3.2	El sistema permitirá al alumno ver sus datos personales y los cursos que ha elegido para el cuatrimestre.	RB4
RF3.3	El sistema permitirá al alumno ver todos los cursos de su sección, que se van a impartir en el cuatrimestre.	RB4
RF3.4	El sistema permitirá al alumno ver todos los cursos que imparte cada investigador de la sección.	RB15
RF3.5	El sistema permitirá al alumno cambiar sus datos personales.	Propuesto
RF3.6	El sistema permitirá al alumno cambiar sus cursos que ha elegido para el cuatrimestre, verificando que los cambios sean dentro de las fechas establecidas y comprobando que no pasen de 4 cursos máximos seleccionados. Los cambios podrán ser realizados siempre y cuando el asesor y el coordinador no hayan realizado cambios en los cursos del alumno.	RB16

RF3.7	El sistema debe permitir al alumno modificar su contraseña pero no su login.	Propuesto
ID	**Requerimientos Funcionales: Servicios al usuario investigador**	**Origen**
RF4	El usuario **investigador académico** podrá realizar los siguientes servicios:	RB10
RF4.1	El investigador académico podrá ver sus datos personales y la información de los cursos que imparte en el cuatrimestre actual de su sección.	RB10
RF4.2	El investigador académico podrá modificar sus datos personales.	Propuesto
RF4.3	El investigador académico podrá ver la lista de alumnos que se inscribieron en los cursos que imparte en la sección y el cuatrimestre actual.	RB10
RF4.4	El investigador académico podrá ver la información de los alumnos que tiene asignados para asesorarlos, así como los cursos que cada alumno tiene asignados para el cuatrimestre actual.	RB10
RF4.5	El investigador académico podrá cambiar los cursos de aquellos alumnos que tienen asignados para asesorarlos. Los cambios sólo podrán realizarse dentro del periodo establecido para este fin, siempre y cuando el coordinador no haya realizado cambios de cursos al alumno.	Propuesto
RF4.6	El SIV debe permitir al usuario investigador académico agregar el contenido de los cursos que va impartir en el cuatrimestre actual.	Propuesto
RF4.7	El SIV debe permitir al usuario investigador académico modificar el contenido de los cursos que imparte en el cuatrimestre actual.	Propuesto
RF4.8	El sistema debe permitir al usuario investigador académico modificar su contraseña pero no su login.	Propuesto
ID	**Requerimientos Funcionales: Servicios al usuario público en general**	**Origen**
RF5	El usuario **público en general** podrá realizar los siguientes servicios:	
RF5.1	El usuario público en general, podrá ver la lista de cursos que se imparten en el DIE.	RB9

Tabla 7.3 Definición de Requerimientos Funcionales del SIV.

En la tabla 7.3, se define los Requerimientos Funcionales del SIV. Cada requerimiento tiene un identificador que corresponde a las primeras letras de Requerimientos Funcionales y un número que corresponde al número de secuencia de los requerimientos. La columna de origen indica de donde se obtuvieron los requerimientos. Estos pueden tener su origen desde la definición básica del cliente, de las entrevistas y cuestionarios (las columnas se indican con las letras E. y C.) o son propuestos por los desarrolladores (indicados por una P).

7.3.3 Definición de Requerimientos No Funcionales del SIV

La tabla 7.4 corresponde a la definición de los **Requerimientos No Funcionales** encontrados en la definición del sistema y son los siguientes:

ID	Requerimientos No Funcionales	Origen
RNF1	El **SIV** proporcionará al usuario una interfaz gráfica a base de menús, ventanas, botones y listas desplegables.	RB3
RNF2	El sistema garantizará la integridad y consistencia de toda la información de las bases de datos.	Propuesto
RNF3	El SIV debe estar disponible las 24 horas del día, para esto debe de contar con un suministro de energía eléctrica constante.	E. y C.
RNF4	El tiempo de aprendizaje para el uso del SIV debe ser máximo de un día para cualquier tipo de usuario, aunque el sistema debe contar con una interfaz gráfica intuitiva.	E. y C.
RNF5	El sistema no debe de exceder de un número mayor a 40 usuarios concurrentes.	Propuesto
RNF6	El navegador del usuario se recomienda que utilice una versión mayor a la 4 para Nestcape Navigator e Internet Explorer.	E. y C.
RNF7	Se debe implementar la manipulación de errores y excepciones en los formularios de las páginas que manejan información.	Propuesto
RNF8	El sistema de contar con interfaces gráficas de usuarios que utilicen los logotipos y colores institucionales.	Propuesto
RNF9	El SIV debe ser visualizado en monitores que cuenten con una resolución de 600X800.	Propuesto
RNF10	El sistema debe contener un proceso que se active de manera automática cada semana para respaldar las bases de datos que se emplean.	Propuesto
RNF11	El login y contraseña deben ser encriptadas para protección de la información de los usuarios.	Propuesto

Tabla 7.4 Definición de Requerimientos No Funcionales del SIV.

7.4 Especificación de los Requerimientos del SIV

La especificación de los requerimientos del SIV proporciona una manera de describir las funciones y restricciones que debe cumplir el sistema.

7.4.1 Definición de Requerimientos Funcionales del SIV usando formas en Lenguaje Natural Estructurado

Otra manera de definir los requerimientos encontrados inicialmente para el SIV, es usando formas definidas en lenguaje natural estructurado, donde se incluye la siguiente información:

1. Una descripción de la función o la entidad a especificar
2. Descripción de entradas y de donde se originan
3. Descripción de salidas y hacia donde van
4. Una descripción de que otras entidades se utilizan
5. Una precondición y una postcondición
6. Efectos colaterales de la operación

Con esta información se pretende detallar aún más los requerimientos con el fin de disminuir las dificultades de interpretación que puedan tener los requerimientos.

SIV: **RF1**

Función: Acceso al SIV.

Descripción: El sistema permitirá dar acceso a usuarios, que serán identificados como: *Alumno* y *Coordinador* mediante un *login* y una *contraseña*, además de un usuario *público en general* que podrá usar el sistema sin necesidad de introducir un *login* y una contraseña.

Entradas: Login y contraseña del usuario.

Origen: Mediante el teclado de la computadora.

Salidas: Validación de usuario o mensaje de error.

Destino: Servidor del SIV (base de datos de accesos).

Requisitos: Servidor del SIV activo, bases de datos de accesos a Alumnos y coordinadores.

Pre-condiciones: Ejecución del servicio www en Internet del servidor SIV.

Post-condiciones: Validación de usuario.

Efectos laterales: Error al tratar de acceder al SIV sin estar registrado en la base de datos de catálogos.

SIV: **RF1.1**

Función: Identificar perfil del usuario.

Descripción: El SIV debe identificar el tipo de usuario y dar servicio de acuerdo a su perfil.

Entradas:

Origen:

Salidas: Visualización de los datos del usuario, como las operaciones que pueden realizar.

Destino: Servidor del SIV (base de datos de accesos).

Requisitos: Usuario este registrado en el SIV.

Pre-condiciones: Acceso al SIV y usuario validado.

Post-condiciones: Visualización de datos y operaciones alcanzable por el tipo de usuario que accede al SIV.

Efectos laterales: Ninguno.

SIV: **RF2.1**

Función: Asignar login y contraseña.

Descripción: El usuario **coordinador** podrá asignar el login y contraseña para los usuarios **alumnos**, **coordinadores** e **investigadores académicos** de la sección a la que pertenece.

Entradas: login y contraseña. La contraseña debe contar con 8 caracteres.

Origen: Teclado.

Salidas: Aviso de confirmación de que la operación es valida.

Destino: Base de datos de usuarios.

Requisitos: Entradas validas de los caracteres para el login y contraseña.

Pre-condiciones: Validación del coordinador.

Post-condiciones:

Efectos laterales: Si el login ya existe enviar un mensaje de advertencia al coordinador.

SIV: **RF2.2**

Función: Agregar un usuario Alumno.

Descripción: El coordinador podrá agregar a un alumno al sistema, esto implica registrar todos sus datos. Después de haber introducido los datos del alumno se debe guardar los cambios para actualizar la base de datos. Cuando se agregue un alumno debe tener un campo de su estado regular, además de asignarle un login y una contraseña de acceso.

Entradas: Datos del alumno[1] e Historial del alumno[2].

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del alumno.

Destino: Base de datos de alumnos en el servidor del SIV.

Requisitos: Entradas validas de los datos para alumno.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Visualización de los datos del alumno y mensajes de guardar información.

Efectos laterales: Si el alumno ya existe enviar un mensaje de advertencia al usuario.

1. Datos del alumno:

Clave del alumno (matricula), nombre, sexo, edad, identificación, domicilio actual, colonia, delegación o municipio, código postal, país, estado y teléfono actual.

2. Historial del alumno:

Clave del alumno (matricula), escuela de procedencia, profesión, título, fecha de titulación, promedio general de la escuela de procedencia, experiencia laboral, último empleo, nombre de la compañía, sección a la que pertenece, tutor asignado, programa de estudio que cursa, fecha de ingreso, fecha de egreso, si tiene baja temporal, causas en caso de tener baja temporal, número de sanciones y causas.

SIV: **RF2.3**

Función: Agregar un curso.

Descripción: El coordinador podrá agregar a un curso al sistema, esto implica registrar toda la información básica referente al curso que se va a registrar en el SIV, sin escribir los detalles del curso que deben ser introducidos por el usuario investigador académico.

Entradas: Datos del curso[3].

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del curso.

Destino: Base de datos de cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para el curso.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Visualización de los datos del curso y mensajes para guardar información.

Efectos laterales: Ninguno.

3. Información del curso:

Clave del curso, sección a la que pertenece, programa de estudio del postgrado, título del curso, nombre del profesor titular, nombre del profesor auxiliar, horario, días en que se imparte, si es de núcleo, número de alumnos inscritos y edificio donde se va a impartir.

Nota.- Los detalles del curso podrán ser agregados por el investigador académico. Los Detalles del curso son: Clave del curso, temas del curso, objetivo que cubre el tema, tiempo de duración y los temas con los que se relaciona.

SIV: **RF2.4**

Función: Agregar un usuario Investigador.

Descripción: El coordinador podrá agregar a un usuario investigador académico al sistema, esto implica registrar todos los datos personales.

Entradas: Datos del Investigador como son: clave, función, fecha de ingreso, si es permanente o temporal, número de empleado, oficina donde labora, teléfono de oficina, extensión, domicilio particular y teléfono particular.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del Investigador.

Destino: Base de datos de Investigador en el servidor del SIV.

Requisitos: Entradas validas de los datos para el Investigador.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Visualización de los datos del investigador y mensajes para guardar información.

Efectos laterales: Si el investigador ya existe enviar un mensaje de advertencia.

SIV: **RF2.5**

Función: Agregar usuario Coordinador.

Descripción: El coordinador podrá agregar a un coordinador al sistema, esto implica registrar todos sus datos en el SIV.

Entradas: Datos del coordinador como son: clave, funciones, fecha de ingreso, si es permanente o temporal, número de empleado, oficina donde labora, teléfono de oficina, extensión, domicilio particular y teléfono particular.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del coordinador.

Destino: Base de datos de coordinador en el servidor del SIV.

Requisitos: Entradas validas de los datos para el coordinador.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Visualización de los datos del coordinador y mensajes para guardar información.

Efectos laterales: Ninguno.

SIV: **RF2.6**

Función: Modificar datos de Alumno.

Descripción: El coordinador podrá modificar los datos de un alumno que se encuentre registrado en el SIV. También podrá cambiar los cursos que el alumno eligió para su cuatrimestre actual.

Entradas: Solo podrán cambiarse algunos datos[4] del alumno y los cursos que tiene asignados para el cuatrimestre actual, esto podrá realizarse siempre y cuando este dentro del periodo establecido para cambios de cursos.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos que pueden cambiársele al alumno.

Destino: Base de datos de alumnos y de inscripción del SIV.

Requisitos: Entradas validas de los datos para alumno y cursos.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de los datos del alumno y mensajes de guardar información.

Efectos laterales: Si se modifico su estado a baja temporal o baja definitiva el alumno no podrá inscribirse a los cursos en lo sucesivo.

Si se le modifico la contraseña al alumno, la próxima vez que trate de entrar al sistema deberá usar esta nueva contraseña.

4. Datos del alumno que pueden modificarse:

Nombre, edad, identificación, domicilio actual, colonia, delegación o municipio, código postal, país, estado y teléfono actual. Además de los cursos a los que esta inscrito en el cuatrimestre actual. Además del Historial los datos que puede cambiar son: título, fecha de titulación, sección a la que pertenece, tutor asignado, programa de estudio que cursa, fecha de egreso, si tiene baja temporal, causas en caso de tener baja temporal, número de sanciones y causas.

SIV: **RF2.7**

Función: Modificar datos del curso.

Descripción: El coordinador podrá modificar los datos[5] de un curso, esto implica cambiar parte de la información referente al curso que se encuentra registrado en el SIV.

Entradas: Datos del curso que pueden ser modificados.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del curso.

Destino: Base de datos de cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para el curso.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de los datos del curso y mensajes para guardar información.

Efectos laterales: Ninguno.

5. Datos del curso que pueden modificarse:

Título del curso, nombre del profesor titular, nombre del profesor auxiliar, horario, días en que se imparte, si es de núcleo, número de alumnos inscritos y edificio donde se va a impartir.

SIV: **RF2.8**

Función: Modificar datos del usuario Investigador.

Descripción: El coordinador podrá modificar los datos de un investigador que se encuentre registrado en el sistema, esto implica cambiar algunos datos[6] referentes al Investigador.

Entradas: Datos del Investigador que el SIV permita modificar.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del Investigador.

Destino: Base de datos de Investigador en el servidor del SIV.

Requisitos: Entradas validas de los datos para el Investigador.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de los datos del investigador y mensajes para guardar información.

Efectos laterales: Ninguno.

6. Datos que pueden modificarse:

Función, fecha de ingreso, si es permanente o temporal, número de empleado, oficina donde labora, teléfono de oficina, extensión, domicilio particular, teléfono particular y los cursos asignados.

SIV: **RF2.9**

Función: Modificar contraseña de usuarios.

Descripción: El coordinador podrá modificar login y contraseña de los usuarios alumno e investigador académico que pertenezcan a su sección y estén registrados en el SIV

Entradas: Cambio de login y contraseña para el usuario.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos que se van a cambiar del usuario.

Destino: Base de datos de usuarios en el servidor del SIV.

Requisitos: Entradas validas de los datos para el usuario.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de los datos del alumno y mensajes de guardar información.

Efectos laterales: En el próximo acceso al sistema el alumno deberá utilizar la nueva contraseña.

SIV: **RF2.10**

Función: Eliminar curso.

Descripción: El coordinador podrá eliminar un curso que se encuentre registrado en el sistema.

Entradas: Elección del curso a eliminar.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del curso que se va a eliminar y confirmación para eliminar.

Destino: Base de datos de cursos en el servidor del SIV.

Requisitos: Selección del curso.

Pre-condiciones: Validación del coordinador y que el curso exista.

Post-condiciones: Actualización de la base de datos del curso.

Efectos laterales: Se debe apreciar esta baja en la lista de curso del alumno que tenga seleccionado este curso para el cuatrimestre actual.

SIV: **RF2.11**

Función: Eliminar usuario Alumno.

Descripción: El coordinador podrá eliminar a un alumno que se encuentre registrado en el SIV.

Entradas: nombre o clave del alumno.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del alumno a eliminar y confirmación de eliminar.

Destino: Base de datos de alumnos en el servidor del SIV.

Requisitos: Alumno exista en la base de datos del SIV.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de la base de datos de alumnos.

Efectos laterales: Se debe reflejar esta baja en los grupos por cursos, es decir, no debe mostrarse este alumno para los cursos que ya había seleccionado.

SIV: **RF2.12**

Función: Eliminar usuario Investigador.

Descripción: El coordinador podrá eliminar a un Investigador que se encuentre registrado en el SIV.

Entradas: nombre o clave del investigador.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los datos del investigador a eliminar y confirmación de eliminar.

Destino: Base de datos de investigador en el servidor del SIV.

Requisitos: Investigador exista en la base de datos del SIV.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de la base de datos de investigadores.

Efectos laterales: Eliminar los cursos en donde es titular.

SIV: **RF2.13**

Función: Establecer periodo de inscripción

Descripción: El coordinador podrá establecer un periodo de inscripción a los cursos del cuatrimestre a los alumnos del SIV. El periodo válido de inscripción comprende 30 días a partir del inicio de cada cuatrimestre.

Entradas: Fecha final en que se pueden inscribir los alumnos a los cursos.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de las fechas limites para inscribirse a cursos.

Destino: Base de datos de periodo en el servidor del SIV.

Requisitos: Que la fecha final sea mayor a la fecha actual.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Actualización de la base de datos de cursos.

Efectos laterales: Mensaje de error si la fecha no es mayor a la actual.

SIV: **RF2.14**

Función: Ver cursos por Alumno.

Descripción: El coordinador podrá ver los cursos que un alumno eligió para su cuatrimestre actual.

Entradas: Nombre del alumno o clave del alumno.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los cursos que el alumno eligió llevar en el cuatrimestre, se muestra el nombre del curso, horario y días en que se imparten.

Destino: Base de datos de alumnos y cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para alumno.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Ninguna.

Efectos laterales: Ninguno.

SIV: **RF2.15**

Función: Ver lista de alumnos por curso.

Descripción: El coordinador podrá ver a los alumnos que se encuentran inscritos en un curso que se imparte en el cuatrimestre actual.

Entradas: Nombre del curso o clave del curso.

Origen: Teclado de la computadora del coordinador.

Salidas: Visualización de los nombres de los alumnos que se encuentran inscritos en el curso y la sección a la que pertenecen.

Destino: Base de datos de alumnos y cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para curso.

Pre-condiciones: Validación del coordinador.

Post-condiciones: Ninguna.

Efectos laterales: Error si no existe el curso a consultar.

SIV: **RF2.16, RF3.4**

Función: Ver los cursos que imparte un investigador

Descripción: El usuario podrá ver todos los cursos que imparte un investigador de su sección.

Entradas: Selección del nombre del investigador.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los cursos que imparte el investigador seleccionado.

Destino: Base de datos de investigador y cursos en el servidor del SIV.

Requisitos: Entrada de la selección valida.

Pre-condiciones: Validación del usuario y que el investigador exista en la base de datos del SIV.

Post-condiciones: Visualización de los cursos que imparte el investigador.

Efectos laterales: Error si no existe el investigador a consultar, o mala selección.

SIV: **RF3.1**

Función: Inscripción a curso.

Descripción: El usuario alumno podrá inscribirse a los cursos registrados en el sistema para el cuatrimestre actual de su sección. El sistema comprobará que la inscripción la realice en el periodo establecido, además de comprobar que el alumno seleccione un máximo de 4 cursos por cuatrimestre.

Entradas: Selección de curso.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los cursos que ha seleccionado para su cuatrimestre.

Destino: Base de datos de alumno en el SIV.

Requisitos: Que el alumno sea regular, el usuario sea valido, validación del periodo establecido para poder inscribirse a los cursos y validación de cuatro cursos como máximo.

Pre-condiciones: Validación del usuario y validación de fecha.

Post-condiciones: Actualización de los datos del alumno, número de cursos seleccionados valido, verificación de cursos de núcleo.

Efectos laterales: Error si sobrepasa el número máximo de cursos tomados por cuatrimestre (4) y advertencia si sobrepasa el número definido de cursos de núcleo.

SIV: **RF3.2**

Función: Ver datos personales.

Descripción: El sistema permitirá al alumno ver sus datos personales y los cursos que ha elegido para el cuatrimestre.

Entradas: Login y contraseña del alumno.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los cursos del alumno.

Destino: Base de datos de alumnos y cursos en el servidor del SIV.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil, alumno regular.

Post-condiciones: Acceso a la base de datos de alumnos.

Efectos laterales: Error si no existe el alumno, esta dado de baja temporal o definitiva.

SIV: **RF3.3, RF5.1**

Función: Ver todos los cursos.

Descripción: El sistema permitirá al alumno ver todos los cursos del DIE, que se van a impartir en el cuatrimestre actual.

Entradas: Login y contraseña del alumno.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los cursos que se van a impartir en el cuatrimestre actual.

Destino: Base de cursos en el servidor del SIV.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil, alumno regular.

Post-condiciones: Acceso a la base de datos de cursos.

Efectos laterales: Error si no existe el alumno, esta dado de baja temporal o definitiva.

SIV: **RF3.5**

Función: Cambiar sus datos.

Descripción: El sistema permitirá al alumno cambiar sus datos personales.

Entradas: Login y contraseña del alumno.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los datos[7] que puede modificar el usuario alumno.

Destino: Base de datos de alumno en el servidor del SIV.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil, alumno regular.

Post-condiciones: Acceso a la base de datos de alumno.

Efectos laterales: Error si no existe el alumno, esta dado de baja temporal o definitiva.

7. Datos del alumno que pueden modificarse:

Nombre, edad, identificación, domicilio actual, colonia, delegación o municipio, código postal, país, estado y teléfono actual. Además de los cursos a los que esta inscrito en el cuatrimestre actual.

SIV: **RF3.6**

Función: Cambiar curso.

Descripción: El sistema permitirá al alumno cambiar sus cursos que ha elegido para el cuatrimestre, verificando que los cambios sean dentro de las fechas establecidas y comprobando que no pasen de 4 cursos máximos seleccionados. Los cambios podrán ser realizados siempre y cuando el asesor y el coordinador no hayan realizado actualmente cambios en los cursos del alumno.

Entradas: Selección del nuevo curso a cambiar.

Origen: Teclado de la computadora del alumno.

Salidas: Visualización del nuevo curso.

Destino: Base de datos de cursos y alumno en el servidor del SIV.

Requisitos: Selección valida de los cursos y último cambio no haya sido efectuado por el coordinador o su asesor.

Pre-condiciones: Validación del alumno como regular, validación del periodo de inscripción, validación del número máximo por cuatrimestre (4) y comprobación de que aun puede efectuar cambio de cursos, es decir que no haya sido efectuado ningún cambio de curso por su asesor o coordinador.

Post-condiciones: Actualización de los datos del alumno, inscripción y mensajes para guardar información.

Efectos laterales: Advertencia si ya no puede realizar cambios debido a la fecha, el número permitido de cambios.

SIV: **RF3.7, RF4.8**

Función: Modificar contraseña.

Descripción: El sistema permitirá al usuario cambiar su contraseña.

Entradas: Login y contraseña del usuario.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los datos del usuario.

Destino: Base de datos de usuario en el servidor del SIV.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil.

Post-condiciones: Acceso a la base de datos de usuario.

Efectos laterales: Error si la contraseña no es valida y el siguiente acceso deben realizarlo con la nueva contraseña.

SIV: **RF4.1**

Función: Ver datos personales.

Descripción: El investigador académico podrá ver sus datos personales y la información de los cursos que imparte en el cuatrimestre actual de su sección.

Entradas: Login y contraseña del investigador académico.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los datos personales y de los cursos que imparte el investigador.

Destino: Base de datos de investigador y cursos en el servidor del SIV.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil.

Post-condiciones: Acceso a la base de datos de investigador y cursos.

Efectos laterales: Error si no es un usuario valido.

SIV: **RF4.2**

Función: Modificar sus datos personales.

Descripción: El investigador académico podrá modificar sus datos personales.

Entradas: Login y contraseña del investigador académico.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los datos personales que puede modificar[8].

Destino: Base de datos de investigador.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil.

Post-condiciones: Acceso a la base de datos de investigador.

Efectos laterales: Error si no es un usuario valido.

8. Los datos que puede modificar son:

Función, fecha de ingreso, oficina donde labora, teléfono de oficina, extensión, domicilio particular y teléfono particular

SIV: **RF4.3**

Función: Ver lista de alumno.

Descripción: El sistema debe permitir a un usuario investigador académico ver la lista de alumnos que se inscribieron en cada curso que imparte en la sección y el cuatrimestre actual.

Entradas: Login y contraseña del investigador académico.

Origen: Teclado de la computadora del usuario.

Salidas: Visualización de los alumnos que asisten a un curso que imparte el coordinador.

Destino: Base de datos de investigador, inscripción, curso y alumno.

Requisitos: Usuario valido y curso se imparta en el cuatrimestre actual.

Pre-condiciones: Validación del usuario y perfil.

Post-condiciones:

Efectos laterales:

SIV: **RF4.4**

Función: Ver información de los alumnos que asesora.

Descripción: El sistema debe permitir al investigador académico ver la información de los alumnos que tiene asignados para asesorarlos, así como los cursos que cada alumno tiene asignados para el cuatrimestre actual.

Entradas: Login y contraseña del investigador académico.

Origen: Teclado de la computadora del usuario investigador académico.

Salidas: Visualización de los datos y cursos de alumnos que tiene asignados para asesorarlos.

Destino: Base de datos de investigador, asesor y alumno.

Requisitos: Usuario valido.

Pre-condiciones: Validación del usuario y perfil.

Post-condiciones:

Efectos laterales:

SIV: **RF4.5**

Función: Cambiar curso a alumnos que asesora.

Descripción: El investigador académico podrá cambiar los cursos de aquellos alumnos que tienen asignados para asesorarlos. Los cambios sólo podrán realizarse dentro del periodo establecido para este fin, siempre y cuando el coordinador no haya realizado cambios de cursos al alumno.

Entradas: Nuevos cursos asignados.

Origen: Teclado de la computadora del investigador académico.

Salidas: Visualización de los datos del alumno y del curso.

Destino: Base de datos de inscripción a cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para el curso.

Pre-condiciones: Validación del investigador académico.

Post-condiciones: Actualización de los datos de la inscripción para el alumno y mensajes para guardar información.

Efectos laterales: Ninguno.

SIV: **RF4.6**

Función: Agregar detalles de un curso.

Descripción: El usuario investigador académico podrá agregar los detalles[9] de un curso, esto implica agregar parte de la información referente al curso que se encuentra registrado en el SIV.

Entradas: Detalles del curso que pueden ser agregados.

Origen: Teclado de la computadora del investigador académico.

Salidas: Visualización de los datos del curso.

Destino: Base de datos de detalles del cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para el curso.

Pre-condiciones: Validación del investigador académico.

Post-condiciones: Actualización de los datos del curso y mensajes para guardar información.

Efectos laterales: Ninguno.

9. Los Detalles del curso son:

Temas del curso, objetivo que cubre el tema, tiempo de duración y los temas con los que se relaciona.

SIV: **RF4.7**

Función: Modificar detalles de un curso.

Descripción: El usuario investigador académico podrá modificar los detalles[10] de un curso, esto implica cambiar parte de la información referente al curso que se encuentra registrado en el SIV.

Entradas: Detalles del curso que pueden ser modificados.

Origen: Teclado de la computadora del investigador académico.

Salidas: Visualización de los datos del curso.

Destino: Base de datos de detalles del cursos en el servidor del SIV.

Requisitos: Entradas validas de los datos para el curso.

Pre-condiciones: Validación del investigador académico.

Post-condiciones: Actualización de los datos del curso y mensajes para guardar información.

Efectos laterales: Ninguno.

10. Los Detalles del curso son:

Temas del curso, objetivo que cubre el tema, tiempo de duración y los temas con los que se relaciona.

7.4.2 Modelos del sistema SIV

Los modelos del SIV proporcionan diferentes enfoques de ver el sistema, agregando detalles en cada perspectiva que ayudan comprender el funcionamiento del sistema. Los modelos que se presentarán son los modelos de datos, modelos de comportamiento y los modelos orientados a objetos.

7.4.3 Modelos de Datos

Los modelos de datos definen la forma conceptual y lógica de los datos procesados por el sistema SIV. El modelo entidad-relación, muestra las entidades de datos y las relaciones entre estas entidades en el diseño de las bases de datos para el SIV.

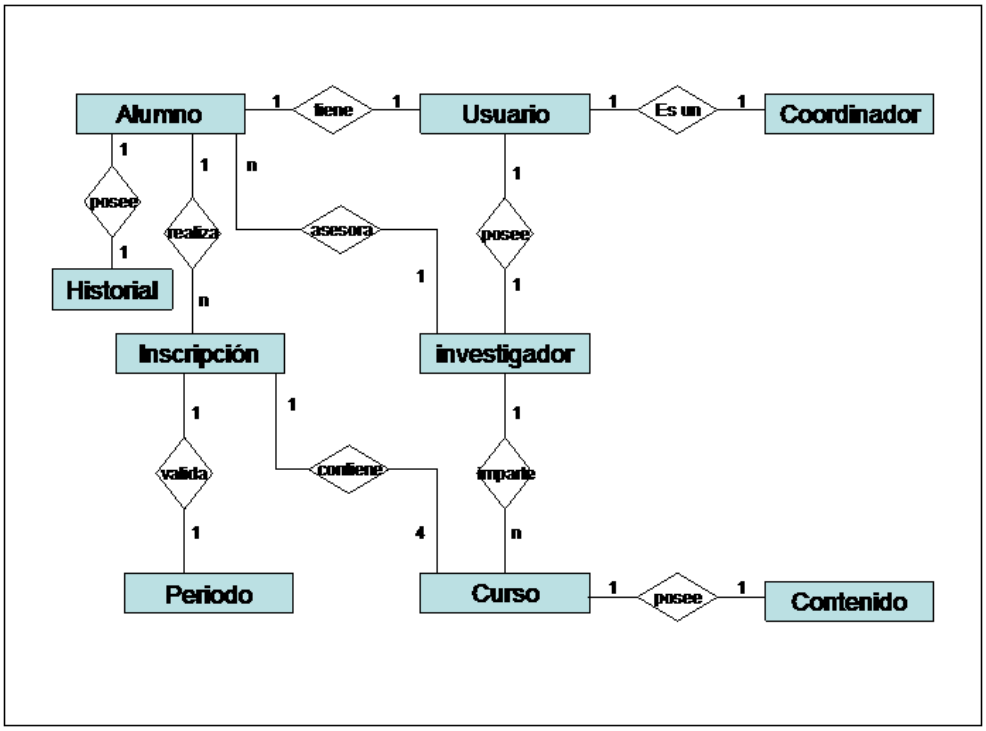


Figura 7.2 Modelo Entidad-Relación del SIV.

En la figura 7.2 la entidad alumno tiene un usuario registrado en el sistema que define su perfil, posee un historial que contiene información complementaria para su ingreso al CINVESTAV, además puede realizar varias inscripciones de cursos en el cuatrimestre siempre y cuando sea un periodo valido. Para un investigador imparte varios cursos, y asesora a varios alumnos.

El coordinador es un usuario del sistema. En una inscripción participan 4 cursos que eligen los alumnos en el cuatrimestre. La tabla 7.5 describe los atributos de cada entidad en el SIV son los siguientes:

Entidad	Atributo	Tipo
Alumno	Clave del alumno	Numérico (entero)
	Nombre	Cadena
	Sexo	Carácter
	Edad	Numérico (entero)
	Identificación	Cadena
	Domicilia actual	Cadena
	Colonia	Cadena
	Delegación o municipio	Cadena
	Código postal	Numérico (entero)
	País	Cadena
	Estado	Cadena
	Teléfono actual	Numérico (entero)
	Clave del investigador asignado (asesor)	Numérico (entero)
Historial	Clave del alumno	Numérico (entero)
	Escuela de procedencia	Cadena
	Profesión	Cadena
	Título	Cadena
	Fecha de titulación	Fecha
	Promedio general de la escuela de procedencia	Numérico (real)
	Experiencia laboral	Cadena
	Último empleo	Cadena
	Nombre de la compañía	Cadena
	Sección a la que pertenece	Cadena
	Programa de estudio que cursa	Cadena
	Fecha de ingreso	Fecha
	Fecha de egreso	Fecha
	Baja temporal	Lógico
	Causa de baja temporal	Cadena
	Número de sanciones	Numérico (entero)
	Última sanción	Cadena
	Causa de sanción	Cadena
Investigador	Clave del investigador	Numérico (entero)
	Nombre	Cadena
	Sexo	Carácter
	Edad	Numérico (entero)
	Número de empleado	Numérico (entero)
	Función	Cadena
	Fecha de ingreso	Fecha
	Permanente o temporal	Lógico
	Oficina donde labora	Cadena
	Teléfono de oficina	Numérico (entero)
	Extensión telefónica	Numérico (entero)
	Domicilio particular	Cadena
	Teléfono particular	Numérico (entero)
	Clave del curso	Numérico (entero)

Tabla 7.5 Descripción de entidades con atributos y su tipo de datos.

Continuación de tabla 7.5 de la descripción de entidades.

Entidad	Atributo	Tipo
Coordinador	Clave del coordinador	Numérico (entero)
	Nombre	Cadena
	Sexo	Carácter
	Edad	Numérico (entero)
	Número de empleado	Numérico (entero)
	Función	Cadena
	Fecha de ingreso	Fecha
	Permanente o temporal	Lógico
	Oficina donde labora	Cadena
	Teléfono de oficina	Numérico (entero)
	Extensión telefónica	Numérico (entero)
	Domicilio particular	Cadena
	Teléfono particular	Numérico (entero)
Usuario-Alumno	Login	Cadena
	Contraseña	Cadena
	Clave del alumno	Numérico (entero)
Usuario-Coordinador	Login	Cadena
	Contraseña	Cadena
	Clave del coordinador	Numérico (entero)
Usuario-Investigador	Login	Cadena
	Contraseña	Cadena
	Clave del investigador	Numérico (entero)
Inscripción	Clave del alumno	Numérico (entero)
	Clave del curso	Numérico (entero)
	Fecha de inscripción	Fecha
	Número cuatrimestre	Numérico (entero)
Periodo	Fecha de inicio	Fecha
	Fecha de término	Fecha
	Número cuatrimestre	Numérico (entero)
Curso	Clave del curso	Numérico (entero)
	Título del curso	Cadena
	Sección a la que pertenece	Cadena
	Programa de estudio	Cadena
	Nombre del profesor titular	Cadena
	Nombre del profesor auxiliar	Cadena
	Horario	Hora
	Días en que se imparte	Fecha
	Es de núcleo	Lógico
	Edificio donde se imparte	Cadena
Detalles del cursos	Clave del curso	Numérico (entero)
	Temas del curso	Cadena
	Objetivo que cubre el curso	Cadena
	Tiempo de duración	Numérico (entero)
	Temas con los que se relaciona	Cadena

Continuación Tabla 7.5 Descripción de entidades con atributos y su tipo de datos.

La tabla 7.5 describe las entidades del sistema SIV, para cada entidad se le definen sus atributos y sus tipos de datos asociados.

7.4.4 Modelos de Comportamiento

Los modelos de comportamiento se utilizan para describir el comportamiento completo del sistema SIV. Se utilizarán los modelos de flujo de datos para modelar el comportamiento de los datos en el sistema SIV. Los modelos de flujo de datos se utilizan para mostrar como fluyen los datos a través de una secuencia de pasos de procesamiento. Los datos se transforman en cada paso antes de moverse a la siguiente etapa, de esta manera permiten al analista comprender el proceso. En la figura 7.3 se muestra el modelo de flujo de datos para el sistema SIV en un nivel alto de abstracción, es decir, solo se muestran los procesos generales sin incluir detalles de operación o restricciones.

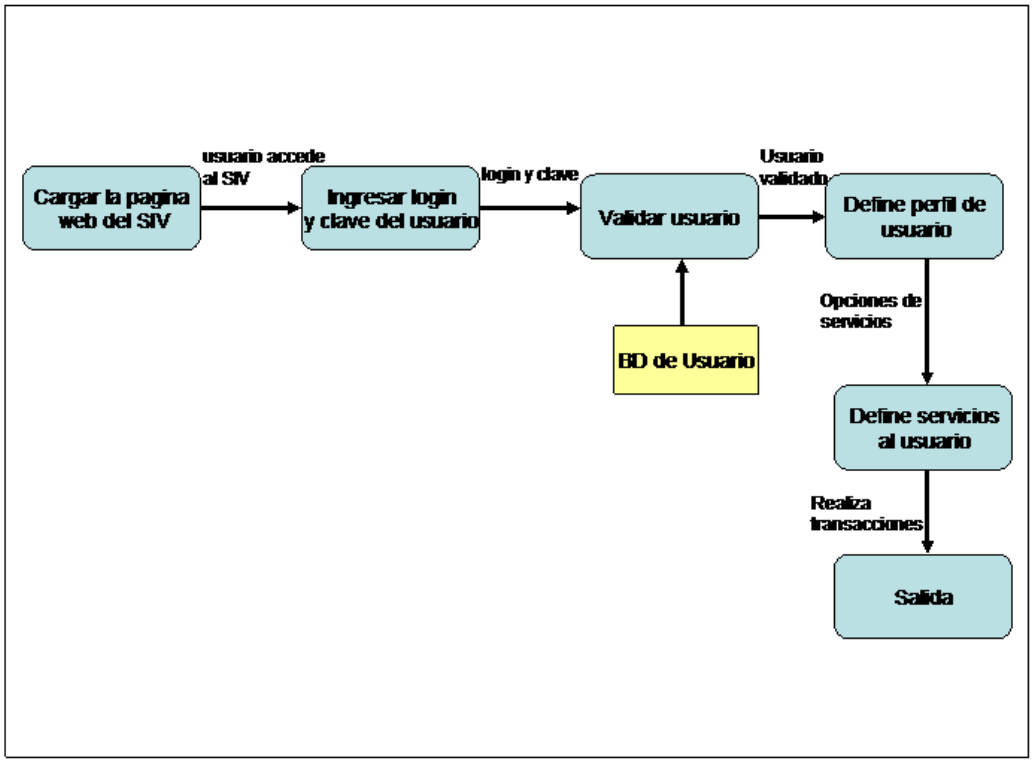


Figura 7.3 Modelo de Flujo de Datos del SIV.

En otro nivel se puede desarrollar la validación de usuarios en el SIV, para cada usuario valido se le asigna un perfil que determina las operaciones que puede realizar. La figura 7.4 muestra como se realiza el flujo de datos para validar un usuario del SIV.

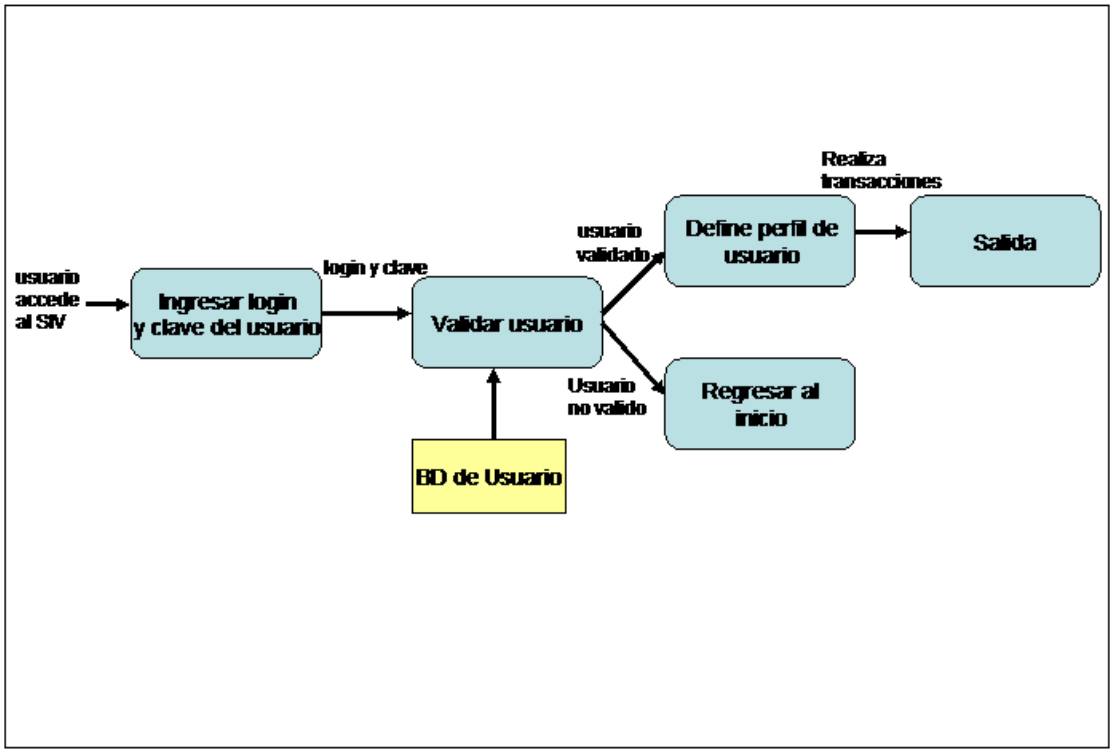


Figura 7.4 Flujo de Datos para el procedimiento de validar usuarios en el SIV.

Para definir el flujo de datos a un nivel mas detallado, en la figura 7.4 se desarrolla el flujo de datos para el usuario alumno, donde se muestran las operaciones que puede realizar. Este diagrama supone que el usuario fue validado y el perfil asignado es el de alumno, entonces comienza a validar al alumno como alumno regular, para poder presentarle los servicios a los que tiene acceso. De este proceso depende el flujo de los datos, es decir, depende de que operación seleccione el alumno el flujo de los datos puede tomar uno de los tres flujos. Finalmente todos llegan a una salida o terminación.

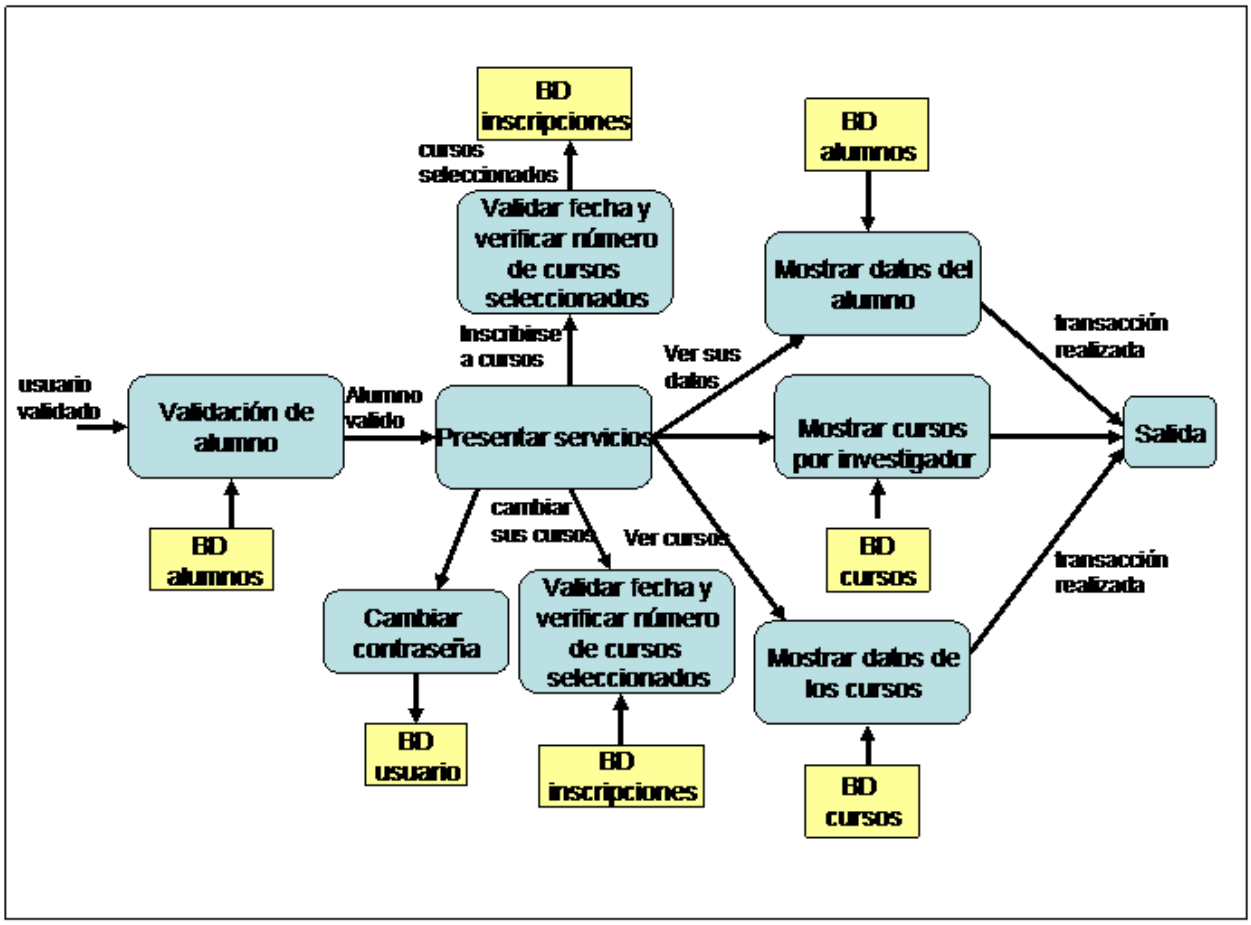


Figura 7.5 Flujo de datos para el usuario alumno del SIV.

Para cada diagrama los rectángulos con esquinas redondeadas representan los procedimientos que corresponden a los procesos que se han de implementar en el sistema, los rectángulos con esquinas cuadradas representan a los almacenes u origen de datos, como son las bases de datos o archivos de información. Las líneas representan los flujos de los datos, es decir como los datos se transforman a través de los procesos. En la figura 7.5 se ilustran los servicios que el usuario alumno podrá utilizar en el sistema. Estos servicios son: la validación del usuario mediante un login y contraseña para definir el perfil y acceso. El alumno podrá inscribirse a los cursos de la sección a la que pertenece, el sistema debe verificar que la inscripción ocurra dentro del periodo establecido y que no pase el límite máximo de cursos por cuatrimestre. El alumno podrá ver sus datos personales y los cursos que tiene asignado para el cuatrimestre, ver los cursos por investigador y podrá modificar su contraseña.

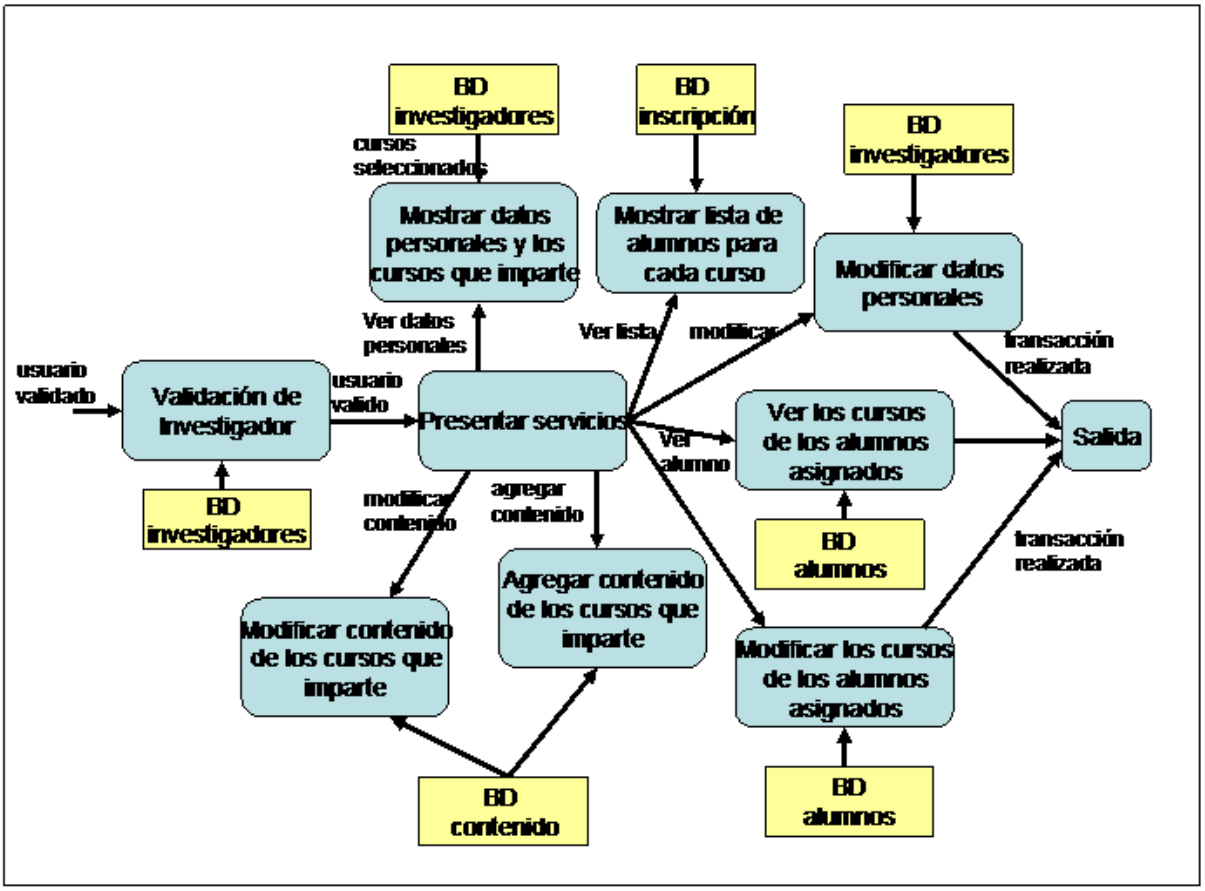


Figura 7.6 Flujo de datos para el usuario investigador académico del SIV.

La figura 7.6 corresponde al diagrama de flujo para el investigador. El usuario investigador entra al sistema SIV y es validado mediante su login y contraseña para que el sistema defina su perfil. Es decir, los servicios que podrá utilizar en el sistema. Estos servicios son: mostrar datos personales y los cursos que imparte en el cuatrimestre actual, ver la lista de alumnos de cada curso, modificar datos personales, ver los cursos asignados a los alumnos que asesora, modificar los cursos de sus alumnos que asesora, agregar y modificar los contenidos de cada curso que imparte.

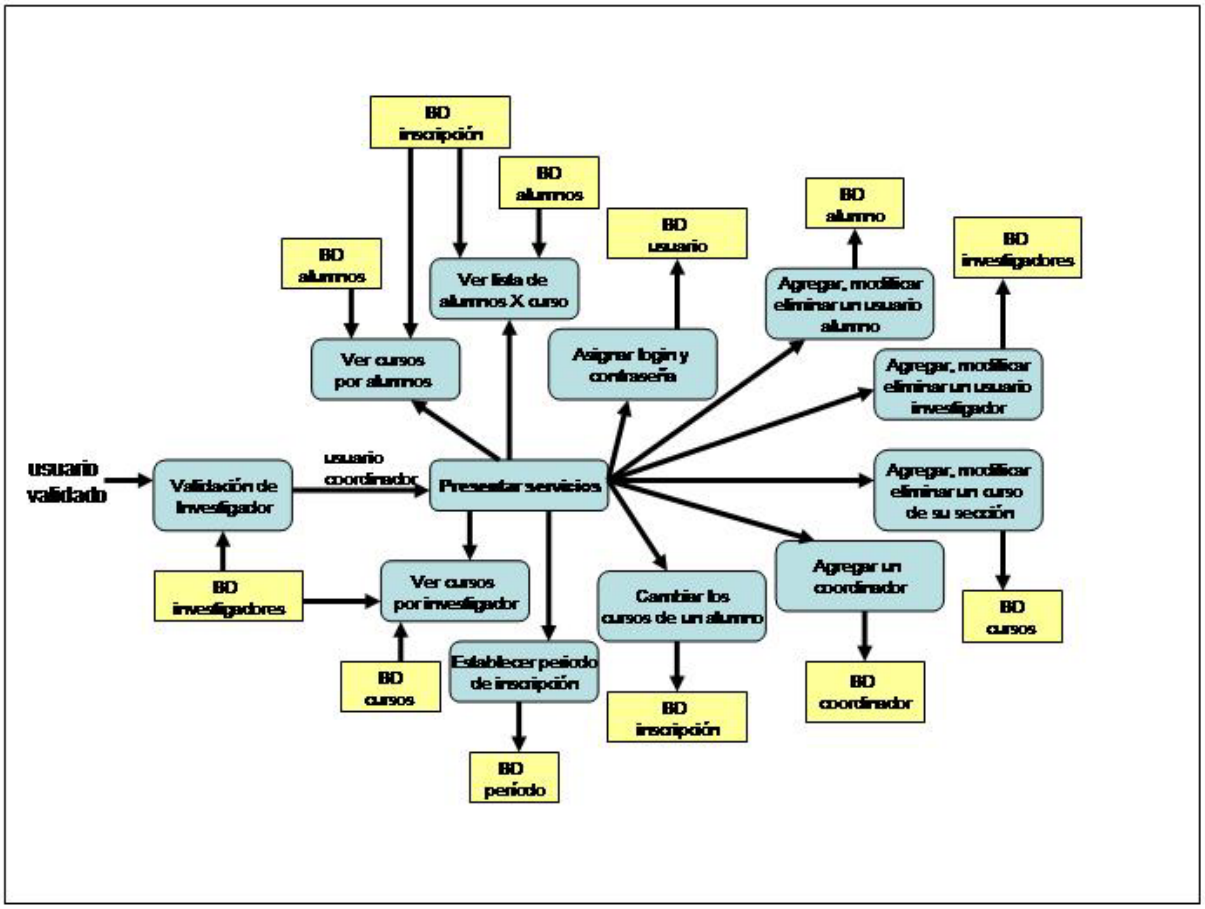


Figura 7.7 Flujo de datos para el usuario coordinador del SIV.

El diagrama de flujo del usuario coordinador se muestra en la figura 7.7 donde el usuario después de que es validado, se define su perfil donde podrá utilizar los servicios asignados. Los servicios asignados son: Asignar y modificar login y contraseña a los usuarios alumnos, investigadores y coordinadores del SIV. Agregar, modificar y eliminar un usuario alumno, investigador y curso de esta forma puede modificar y actualizar las bases de datos de alumno, investigador y curso respectivamente. Agregar un usuario coordinador, establecer periodo de inscripción, ver los cursos por investigador, ver la lista de alumno por cada curso que se imparte en su sección y ver y modificar los cursos que cada alumno tiene asignado.

7.4.5 Modelos Orientados a Objetos

Los modelos de objetos desarrollados durante el análisis de requerimientos se utilizan para representar los datos del sistema y su procesamiento, son útiles también para mostrar la manera en que las entidades del sistema se clasifican y componen de otras entidades.

Los Casos de Uso son parte de los modelos de objetos y se utilizan para demostrar la interacción que existe entre el usuario y el sistema. Cada usuario es representado mediante un actor (representado pos un diagrama de muñeco) que realiza una tarea o caso de uso en el sistema. Los casos de uso se representan mediante una burbuja y una flecha que une un actor con un caso de uso. En la figura 7.8 se representan los actores alumnos y público en general para los servicios del SIV.

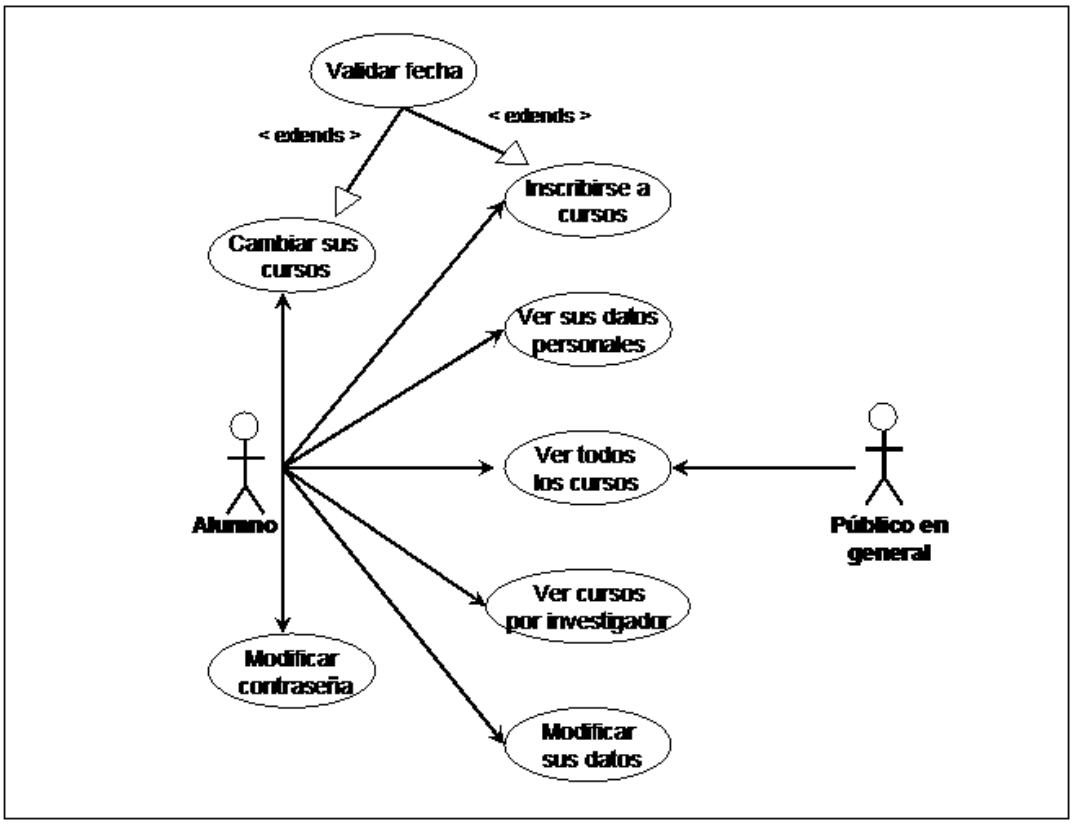


Figura 7.8 Caso de uso para el actor alumno y público en general.

Los casos de uso representan las funciones u operaciones que cada actor puede realizar dentro del sistema SIV. En la figura 7.8, el caso de uso *inscribirse a cursos* utiliza otro caso de uso (extends) para validar la fecha de inscripción. Esto se refiere a comprobar que cuando se realice la inscripción a los cursos, este en las fechas validas permitidas.

El modelo de casos de uso para el usuario coordinador es más complejo debido a la especificación del SIV donde el coordinador es el responsable de ejecutar la mayoría de las funciones y servicios del SIV. En la figura 7.9 se muestran todas las operaciones que podrá realizar un actor coordinador en el sistema.

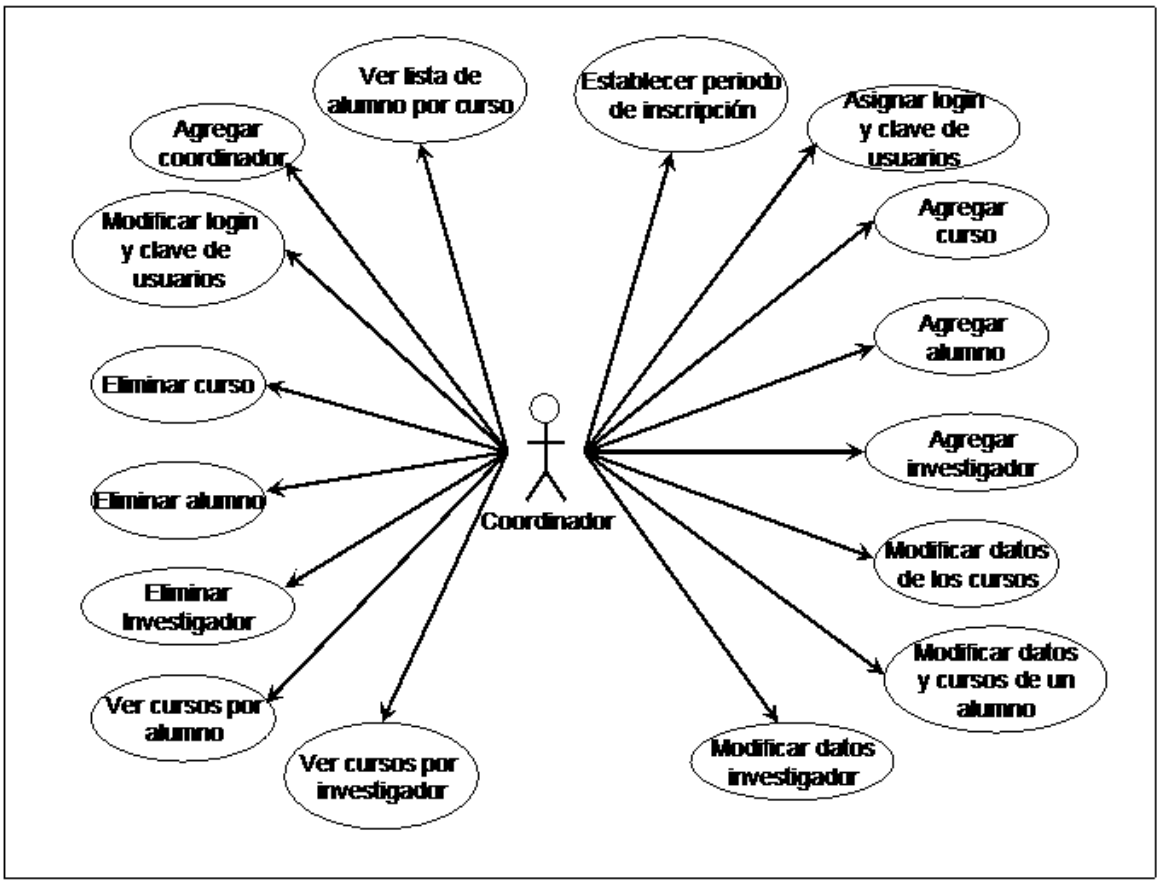


Figura 7.9 Modelo de Casos de uso para el actor Coordinador en el SIV.

El modelo de casos de uso para el actor investigador del SIV se muestra en la figura 7.10. Los servicios que puede realizar un actor investigador académico del CINVESTAV-IPN son ver la lista de alumnos inscrito en un curso que imparte, ver sus datos, ver la lista de alumnos que tiene asignado para darle asesoria y ver todos los curso que imparte.

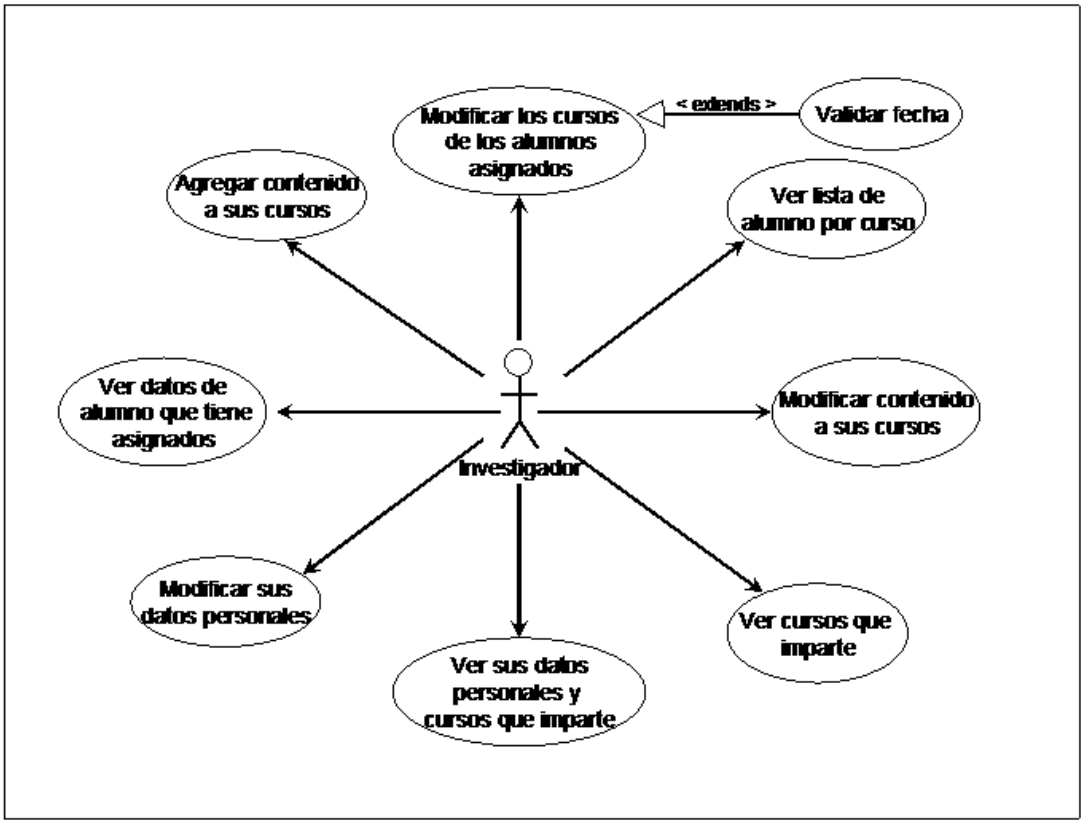


Figura 7.10 Modelo de Casos de uso para el actor investigador académico del SIV.

Los diagramas de clases describen los diferentes tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos. El diagrama de clases simplificado para el sistema SIV se muestra en la figura 7.11

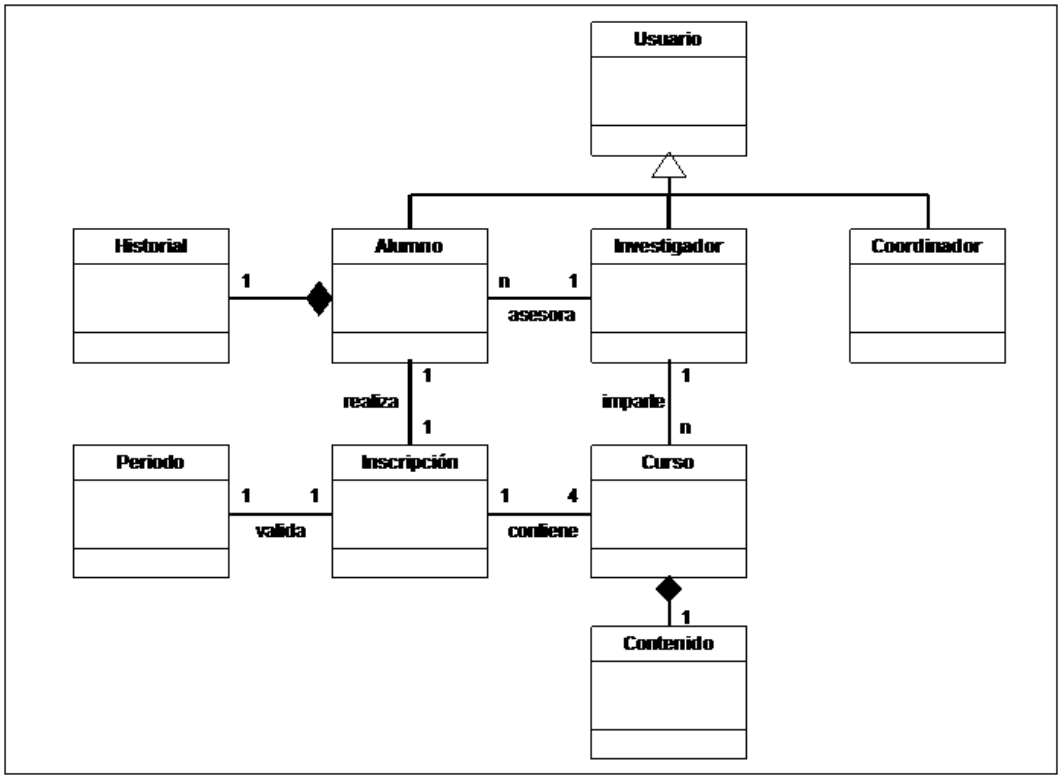


Figura 7.11 Diagrama de clases para el SIV.

El la figura 7.11 se encuentran definidas nueve clases que representan las estructuras de los objetos que intervienen en el SIV. No se definieron los atributos de las clases ni sus servicios u operaciones por la dimensión del diagrama. Los atributos son los mismos que están definidos en la tabla 7.5. Las clases se representan utilizando un rectángulo, en la parte central superior se define el nombre de la clase, en la parte de en medio se escriben todos los atributos de la clase, y en la parte baja se definen las operaciones que realiza esa clase. Las relaciones o asociaciones están definidas por medio de una línea que une a dos clases. Por ejemplo, La clase usuario es una generalización de las clases alumno, investigador y coordinador. La clase historial tiene una asociación de composición con la clase alumno, es decir, la clase historial es parte de la clase alumno. La clase investigador y la clase alumno poseen una asociación simple, y se lee como un investigador asesora a varios alumnos. Hasta ahora se han definido los diagramas de casos de uso que describen las tareas que el sistema debe ejecutar y los diagramas de clases que definen las clases implicadas y las relaciones entre ellas.

Los diagramas de secuencias muestran como el sistema realiza un caso de uso, o un escenario en particular de un caso de uso, donde los actores, objetos y enlaces entre ellos interactúan para ejecutar una tarea. En la figura 7.12 se muestra el diagrama de secuencia para la inscripción a cursos de un alumno.

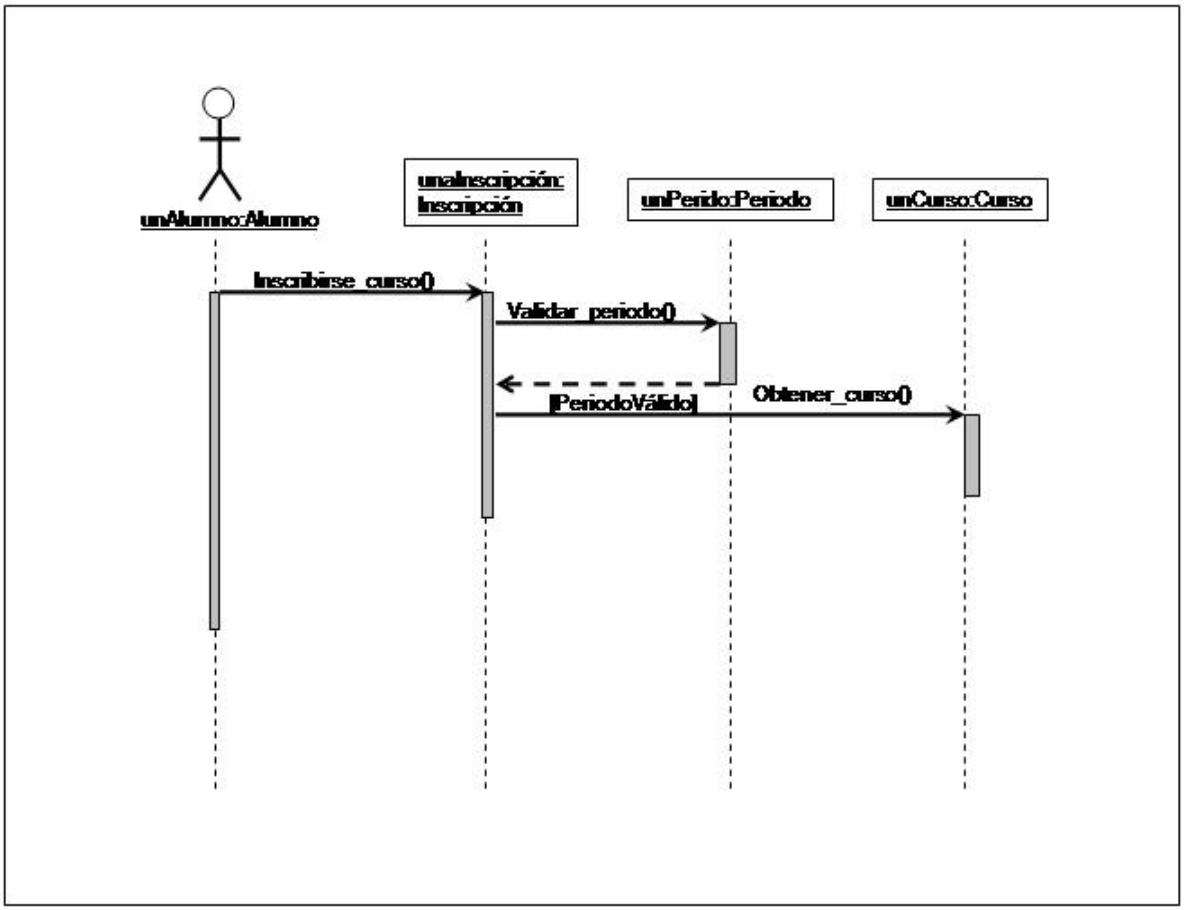


Figura 7.12 Diagrama de secuencia para la inscripción a cursos por un alumno.

En la figura 7.12 el objeto unAlumno realiza la tarea de inscribirse a un curso enviando el mensaje inscribirse_curso() al objeto unaInscripción, este objeto envía un mensaje validar_periodo a la clase Periodo y se crea el objeto unperiodo, retornando un valor verdadero si la fecha es valida para que después el objeto unainscripción envié un mensaje de Registrar_curso() a la clase Curso creando un objeto incurso. Las líneas punteadas verticales que parten de los objetos se les conocen como **líneas de vida**, y corresponden a la duración del objeto en la interacción.

Las flechas punteadas son los mensajes que los objetos se envían, se omiten las líneas de retorno a menos que sea necesario para aclaración del diagrama.

Otro ejemplo de un diagrama de secuencia es el de la figura 7.13, donde un alumno modifica un curso en su inscripción.

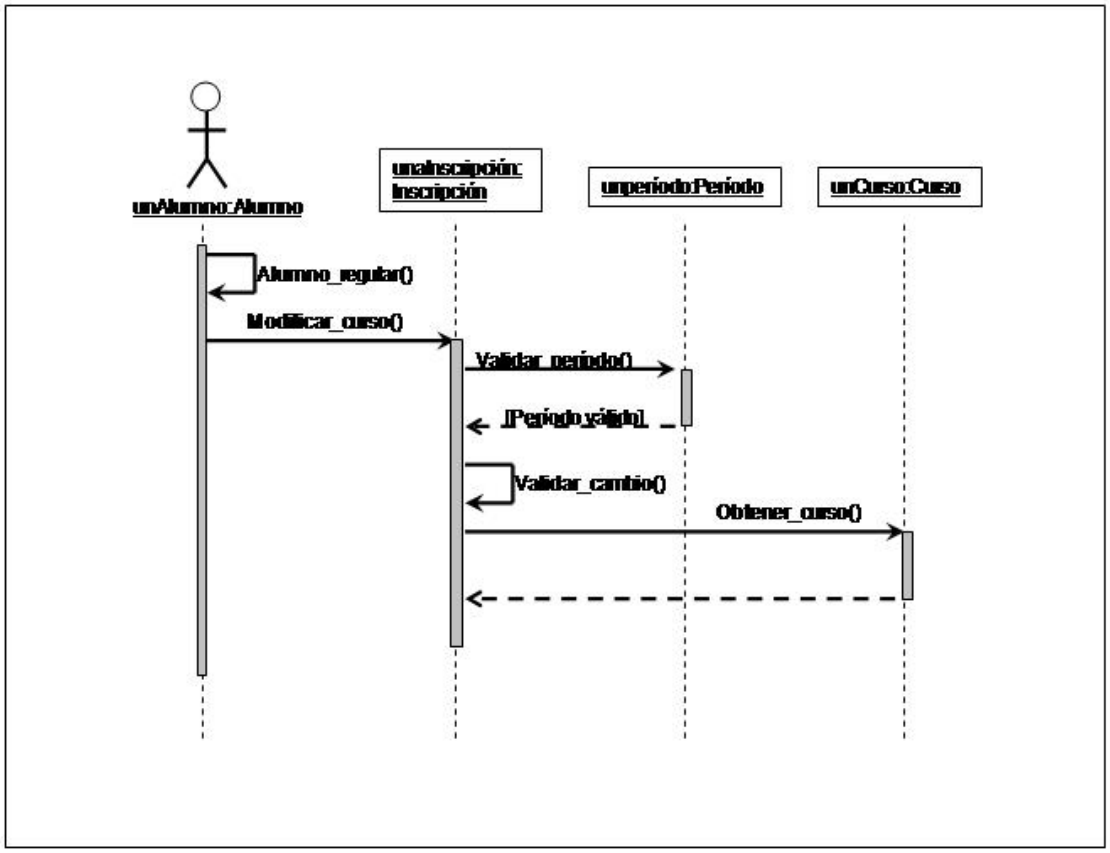


Figura 7.13 Diagrama de secuencia para modificar un curso por un usuario alumno.

En este escenario (Figura 7.13), se lleva a cabo la comprobación del objeto unAlumno sea un alumno regular antes de poder modificar los cursos de su inscripción, después mediante el mensaje Modificar_curso() se crea el objeto una inscripción que comprueba que la fecha sea válida en la clase Periodo. Si la fecha es válida, se envía un mensaje así mismo para comprobar que puede realizar cambios en sus cursos, es decir, verifica que su asesor o coordinar no le ha hecho algún cambio de lo contrario ya no podrá modificar sus cursos, esto es de acuerdo a la especificación del requerimiento RF3.6. Después de validar el cambio envía el mensaje Obtener_curso() a la clase Curso para crear un objeto unCurso y obtener la clave del curso que será regresada para modificarlo en su inscripción.

7.5 Evolución del SIV

Considerar funciones bajo las cuales el sistema puede actualizarse es anticiparse al cambio, sin que afecte el funcionamiento del sistema. Cualquier mejora o agregación en la funcionalidad del sistema implica directamente en el crecimiento del volumen de la información, por lo que cualquier implementación debe ser evaluada sin que esto implique riesgos en el desempeño y en la utilización del sistema. Los principales puntos que se están contemplando son los siguientes:

1. **Políticas y procedimientos para el respaldo de la información**. En este sentido se contempla en primer lugar la seguridad que debe tener nuestro cliente al conservar la integridad de su información. Independientemente de factores en el ambiente que puedan representar ser elementos de riesgo. En este punto podemos hablar de fallas de corriente eléctrica, información que pueda dañarse, equipos dañados, virus, etc. Todo lo que pueda dañar su información.

   Se han tomado en cuenta estos riesgos y la solución en primera instancia que se propone es crear políticas y mecanismos de respaldo de información. Para los mecanismos de respaldos de información se contemplan los siguientes puntos:

   - Generación de documentos donde se explique paso a paso la realización técnica del respaldo de la información. Estos documentos recomendamos incluirlos en los procedimientos de la empresa.
   - Integración de un equipo de respaldo de información.
   - Calendarización para la ejecución de dichos procedimientos.
   - Mecanismos de revisión.

2. **Instalación de seguridad en el nivel de protocolos de red con un certificado de seguridad Secure Socket Layer (SSL)**. Para este punto se propone que a futuro la información se pueda manejar de manera segura en el nivel del protocolo. Utilizando https (http seguro), esto se logra con la instalación de un certificado de seguridad, normalmente es Secure Socket Layer. En este punto el software que se sujeta a cambios es el servidor de páginas Web. Lo cual resulta benéfico porque el sistema no sufre cambios es su desarrollo y estructura interna. El resultado es que viaja la información por la red de Internet de manera segura.

3. **Manejo de bitácoras donde se refleje información estadística de desempeño del sistema**. Es importante saber el tipo de servicio que estamos entregando con el sistema. Con elementos que puedan evaluar el desempeño del sistema y se puedan tomar como referencia para mantenimientos preventivos, correctivos, descubrimiento de fallas (en caso de existir) y elementos de toma de decisión para futuros proyectos.

4. **Crecimiento y actualización de hardware**. Algo que siempre es seguro es que la tecnología avanza rápidamente, los volúmenes de información van a crecer con el paso del tiempo. Lo cual siempre implica un crecimiento en la plataforma de hardware es decir siempre tendremos que incrementar el tamaño del disco duro, y la memoria como mínimo.

5. **Crecimiento y actualización del software**. Agregar nuevas funciones al sistema, es algo común en todo software, pero debe considerarse que estas implementaciones no afecten las estructuras definidas de los datos, la especificación de los requerimientos esta planeada para soportar cambios de los requerimientos, pero estos cambios deben evaluarse para no afectar a otros requerimientos relacionados.

6. **Propuesta para la integración de un equipo espejo para hacer el sistema redundante**. Este punto lo estamos manejando como propuesta ya que es una sugerencia que pudiera tomar en cuenta el cliente sin que se sienta obligado a aceptarla. La idea es implementar un sistema igual al que estamos proponiendo para que tenga las funciones de un equipo espejo con balanceo de carga. Un sistema redundante tiene un alto porcentaje en la garantía de un servicio continuo. Ya que distribuye el número de peticiones que recibe el sistema. Evita que el sistema este fuera de servicio por falla del servidor. Tiene un tiempo de respuesta más alto y eficiente. Debido a que se cuenta con un respaldo que atiende las peticiones. Para esta situación se tiene que manejar una propuesta similar ya que se integrarían algunos cambios en la parte de red para generar la redundancia, considerando un equipo nuevo y otros factores.

7.6 Glosario

Conceptos	Definición
Administrador	Persona encargada de la operación, mantenimiento y administración del sistema SIV.
Alumno	Personas estudiantes que se encuentran legalmente escritas en una escuela, en este caso, en el CINVESTAV-IPN.
Base de Datos	Deposito de información relacionada y estructurada para su manipulación y uso.
BD	Véase Base de Datos.
Casos de Uso	Representa la interacción entre un usuario y el sistema. Este término pertenece al enfoque orientado a objetos.
CINVESTAV-IPN	Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional.
Clases (Diagramas de)	Es la representación de los tipos de objetos que hay en el sistema, este término es empleado en el enfoque orientado a objetos.
Cliente	Persona física que solicita la manufactura de un sistema y se encarga de realizar el contrato con el equipo de desarrollo. Generalmente se le denomina de esta forma porque él es el que realiza el pago del sistema.
Colegio de Profesores	Se le denomina de esta manera al conjunto de profesores investigadores de cada sección del departamento de Ingeniería Eléctrica.
Contraseña	Es una clave compuesta de letras y número que identifica de manera única a un usuario del sistema, y mediante ésta le permite el acceso.
Coordinador o Coordinador académico	Personal del Departamento de Ingeniería Eléctrica que se encuentra asignado como responsable de una sección de este departamento.
Curso	Ocurrencia donde se lleva a cabo el proceso de enseñanza-aprendizaje, de una materia.
DER	Documento de Especificación de Requerimientos, en el se encuentran definidos todos los requerimientos del sistema a construir.
Desarrollador	Ingeniero de Software que se encarga de la manufactura del sistema, puede tener cualquier especialidad, como por ejemplo analista, diseñador, arquitecto, programador, probador o entrenador.
Diagrama de Flujo de Datos	Es una secuencia lógica de pasos que se siguen para resolver un problema, utilizando terminologías y diagramas.
DIE	Departamento de Ingeniería Eléctrica.
Interfaz	Medio de comunicación que se da entre dos entidades.
Internet	Red de redes de computadoras, con el fin de compartir recursos e información. Red Internacional de computadoras.
Internet Explorer	Software que permite la visualización de página de información de Internet, propiedad de Microsoft Corporation.
Login	Nombre descriptivo de un usuario que le permite el ingreso al sistema y define las operaciones a las que tiene derecho.
Modelos	Son representaciones de los sistemas.
Netscape Navigator	Software que permite la visualización de página de información de Internet.
PDL	Lenguaje de Descripción de Programa, se utiliza para especificar la operación y flujo de datos para un sistema de software.
Personal Administrativo	Personal del Departamento de Ingeniería Eléctrica que se encarga de las tareas administrativas.
Servidor del SIV	Equipo de cómputo donde reside el software y las bases de datos del SIV.
SIV	Sistema de Inscripción Virtual de alumnos del Departamento de Ingeniería Eléctrica del CINVESTAV.
Stakeholder	Cualquier usuario involucrado en el SIV.
Usuario	Persona que utiliza un servicio, proceso o sistema de cómputo.

Capítulo 8

Conclusiones y Trabajo Futuro

8.1 Conclusiones

La Ingeniería de Requerimientos como parte del proceso de desarrollo de software, es un puente importante hacia las otras etapas, como son el diseño, la implementación, la validación y el mantenimiento. Esto significa que una descripción completa de los requerimientos garantiza el desarrollo de un buen producto final. Sin embargo, la obtención y especificación de los requerimientos son dos situaciones problemática que todo ingeniero de software enfrenta. Por un lado se busca resolver las diferencias de comunicación, culturales y técnicas que enfrentan los usuarios y los desarrolladores en el afán de lograr obtener las necesidades y restricciones que definirán el sistema; y por el otro, la búsqueda de técnicas, métodos, metodologías y herramientas que permitan obtener una especificación de los requerimientos entendible tanto para los clientes y usuario como para los ingenieros involucrados en el diseño.

En este trabajo de investigación se revisaron diversas técnicas y metodologías que existen para el proceso de Ingeniería de Requerimientos, se realizaron análisis para seleccionar la técnica o técnicas para aplicarlas en el desarrollo del caso de estudio (SIV), sin embargo, en el estudio de las metodologías, se obtuvieron resultados no satisfactorio, es decir, ninguna de las metodologías que se analizaron no comprendían todas las actividades que se definieron en el proceso de la Ingeniería de Requerimientos. Por lo que se definió una metodología que cubriera todas las actividades y tareas del proceso de la Ingeniería de Requerimientos, así como las técnicas específicas para lograr realizar cada actividad, considerando las características del sistema, del

entorno y de los usuarios. Además se propuso un esquema del documento de requerimientos tomando como base el estándar definido por la IEEE.

Como experiencia obtenida se puede afirmar que es difícil proponer una técnica, método o metodología que cubra universalmente a los sistemas en desarrollo, esto se debe a que cada sistema tiene características particulares que lo hacen diferente a los demás, sin importar que sean del mismo ambiente, además de las características de los usuarios y clientes. Por lo que las técnicas, metodologías y herramientas aquí usadas no necesariamente van a obtener los mismos resultados en su aplicación para otro sistema, más bien, son una guía y servirán como referencia en el desarrollo de la Ingeniería de Requerimientos de cualquier sistema.

8.2 Trabajo Futuro

El campo de los requerimientos del sistema es un área muy extensa y hay mucho trabajo por hacer, por ejemplo, no existen herramientas de software que se realicen automáticamente la validación de los requerimientos mediante los atributos de calidad presentados en esta investigación. No existen herramientas que vinculen los requerimientos funcionales con los requerimientos no funcionales, donde se ilustren la importancia y el efecto que tendrán estos vínculos. Aún no existe la manera automatizada de conocer cuando los requerimientos están completos en el documento de requerimientos, este proceso actualmente se lleva a cabo mediante sesiones donde los usuarios, el cliente y el equipo de desarrollo realizan las revisiones pertinentes.

Apéndice A

Entrevistas

Entrevistas a usuarios del SIV

Siguiendo esta línea de obtención de información para la definición del contexto del sistema SIV, se realizaron entrevistas a usuarios del actual sistema de inscripción, el protocolo de la entrevista fue el siguiente:

1. Presentación de las personas que integran el equipo de desarrolladores
2. Objetivo de la entrevista
3. Sección de preguntas
4. Comentarios adicionales
5. Despedida

Como es obvio, no se explican los puntos uno, dos y cinco del protocolo de las entrevistas, sólo describiremos las secciones de mayor importancia. En la sección de preguntas, se realizaron las siguientes:

Preguntas en las entrevistas.

1. *¿Cómo se lleva a cabo actualmente el proceso de inscripción a cursos que imparte su sección del Departamento de Ingeniería Eléctrica?*
    a. *Es manual.*
    b. *Automatizado (parcial o total).*
    c. *Mixto.*
2. *¿Cuál es el procedimiento del proceso de inscripción a cursos por un alumno?*
3. *¿Cómo está estructurada la selección de cursos de su sección?*
4. *¿Existen políticas o restricciones a los alumnos para la inscripción a los cursos que se imparten en su sección?*
5. *¿Qué requisitos se deben cumplir para ser alumno de la sección?*
6. *¿Qué requisitos se deben cumplir para ser alumno regular?*

7. *¿Cómo son almacenados los expedientes de los alumnos?*
8. *¿Qué información se le solicita al alumno para integrar su expediente?*
9. *Si el sistema de inscripción a cursos de su sección se automatizara, ¿Estaría usted de acuerdo? ¿Por qué y qué beneficios obtendría?*

Las entrevistas realizadas se llevaron a cabo únicamente en la sección de computación, y se obtuvo la siguiente información:

Las personas entrevistadas fueron:

Usuario entrevistado	**Cargo**
Sofía Reza	Secretaria encargada de Control Escolar de alumnos, en la Sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN
Dr. Jorge Buenabad Chavez	Coordinador Académico de la sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN

Información obtenida de la Sección de Computación

De la información obtenida se tomaron únicamente los puntos de importancia para la identificación de requerimientos y se describe en el orden de las preguntas realizadas a los usuarios del SIV. La información obtenida es la siguiente:

1. *El proceso de inscripción a cursos de la sección, actualmente es mixto, ya que se combina un proceso manual y automatizado, como se describe en lo sucesivo.*
2. *El alumno selecciona los cursos que va a tomar en el cuatrimestre actual en coordinación y aprobación de su asesor asignado. (Procedimiento manual)*
    - *El alumno entrega una hoja de inscripción firmada por el asesor a la secretaria encargada del control escolar de la sección (Sofía Reza).*
    - *Esta información es vaciada a una base de datos que se tiene en la coordinación académica de la sección y manipulada por la secretaria, para la formación y control de grupos, y generación de boletas.*
    - *Se almacena la hoja de inscripción a cursos en el expediente de cada alumno, integrándola en una carpeta de expediente, en un archivero que controla la secretaria de control escolar de la sección.*
    - *Cabe señalar que un Académico, funge como asesor de un numero de alumnos de cada generación, de acuerdo al interés de especialización de cada alumno y el perfil del académico, esta asignación es por lo menos, durante el primer año escolar.*
    - *La función del asesor de cada alumno es auxiliarle y orientarlo en la selección de cursos de acuerdo a su perfil.*
    - *A partir del inicio del cuatrimestre tiene 30 días para solicitar cambio de cursos, previa justificación y aprobación tanto del asesor asignado al alumno como del coordinador académico.*
    - *La selección realizada de cursos, es enviada a Control Escolar del CINVESTAV-IPN, para su integración al expediente general.*
3. *El coordinador académico y el Colegio de Profesores realizan la programación de cursos de acuerdo al reglamento y al programa de la maestría.*

- *Esto se realiza de acuerdo a las inquietudes manifestadas por los alumnos, el colegio de profesores y el coordinador programan los cursos que pueden impartir en el presente cuatrimestre.*
- *Se toman en cuenta también el programa académico de postgrado de cada sección y apegado al reglamento.*

4. *Las políticas y restricciones se encuentran especificadas en el reglamento de cada sección (se anexa copia del reglamento interno de la sección, en el apéndice).*
    - *Sólo puede elegir cuatro cursos por cuatrimestre, en cualquier otro caso previa justificación y autorización del asesor y coordinador académico.*
    - *La inscripción a cursos es válida dentro de las fechas señaladas.*
    - *Cursar cuatro curso de núcleo y los demás de su área de especialización.*
5. *Haber cumplido y efectuado el proceso de admisión de la sección (se anexa el procedimiento en el apéndice)*
6. *Estar actual y legalmente inscrito a la sección, no haber causado baja temporal o definitiva, no estar suspendido (Verificar reglamento interno de la sección, copia anexada en el apéndice)*
7. *El expediente del alumno, es información de interés para la sección:*
    - *Los expedientes son almacenados individualmente en carpetas de papel con un identificador, para control y administración de la secretaria de control escolar de la sección.*
    - *Las carpetas se almacenan en archiveros, ordenados por generación de cada sección.*
    - *El acceso a los expedientes es mediante la secretaria de control escolar interno a la sección*
8. *y consiste en: Solicitud de admisión, Fotografías, Acta de examen final o título, Certificado de calificaciones, Dos cartas de recomendación de investigadores, currículo, Constancias de cursos, congresos, seminarios, etc., Acta de nacimiento, carta de pasante y CURP.*

- *Además se integra cada cuatrimestre la inscripción a cursos.*

9. *Absolutamente de acuerdo, un proceso automatizado simplifica el procedimiento actual, ahorra tiempo y trabajo, se podría realizar de manera remota, reduce la tasa de errores, se tiene la información al alcance y en cualquier momento de manera rápida y eficiente.*

Comentarios adicionales.

- *La información de cada sección es entregada a Servicios Escolares del Cinvestav–IPN. Después de este proceso, la información de todos los alumnos es entregada a Servicios Escolares del CINVESTAV-IPN, para su selección y tratamiento, y que ya no corresponde a las secciones del departamento de Ingeniería Eléctrica.*
- *Se proporcionaron documentos que respaldan la información declarada y son integrados en los apéndices, para futuras referencias o validación de la información.*

En el resultado de las entrevistas encontramos datos de interés para la definición de los requerimientos del SIV, y para no ser redundantes se compararon con los datos ya encontrados y sólo se muestran aquellos nuevos datos involucrados con el SIV:

De los tipos de usuarios y entidades internas:

- Secretaria de control escolar.
- Académico, también llamado asesor.

De los servicios del SIV:

- Permitir a alumnos seleccionar un máximo de 4 cursos por cuatrimestre
- Asegurar que el alumno se inscriba en los cursos únicamente en las fechas establecidas, no sobrepasando los 30 días que tiene para cualquier cambio de curso

Como se observa se encontraron nuevos datos y información que tiene relevancia para el funcionamiento del sistema SIV.

Aplicaciones de cuestionarios a usuarios del SIV

De los cuestionarios que fueron aplicados se obtuvo información para la identificación de los requerimientos tanto de usuarios como requerimientos del sistema. Las preguntas elaboradas en los cuestionarios fueron:

1. *Se desea particularmente que se construya una interfaz gráfica amigable y eficiente.*
    - ***¿Cómo calificaría amigable?***
    - ***¿Para usted que sería eficiente?***
2. *El sistema SIV debe permitir a un alumno elegir los cursos que desea tomar en el cuatrimestre de la sección de departamento de Ingeniería Eléctrica a la que pertenezca.*
    - ***¿Bajo que condiciones se pueden elegir materias?***
    - ***¿Hay algún mapa curricular que deba seguir?***
3. *Sólo podrá hacer uso del sistema SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares (que no estén dados de baja) en alguna sección del DIE.*
    - ***¿Cómo se sabe que alumnos están regulares?***
4. *Todos los accesos al sistema SIV deberán realizarse desde una interfaz gráfica accesible desde Internet.*
    - ***¿Alguna característica en específico?***
5. *El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materia.*
    - ***¿Su tutor académico no tendría que autorizarlo?***
    - ***¿Cuál es la política de su departamento?***
6. *Así mismo, el sistema permitirá la creación y modificación de bases de datos conteniendo información de los cursos y de los alumnos inscritos en cada sección del DIE del CINVESTAV-IPN.*
    - ***¿Creación y modificación de bases de datos?***
    - ***¿Cuál es la idea detrás de esto?***

7. *El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.*
    - ***¿Quién será el encargado de dar estas claves?***
    - ***¿Cómo se notificará a cada alumno de su clave?***
    - ***¿Se permitirá que cada alumno pueda personalizar su clave?***
8. *Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrán acceder la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.*
    - ***¿Quién será el encargado de dar éstas claves?***
    - ***¿Cómo se le notificará su clave?***
    - ***¿Qué se planea hacer?***
9. *Se pueden considerar las siguientes vistas al sistema:* ***Público en general*** *(los cuales solo pueden consultar datos de los cursos).* ***Alumnos*** *del CINVESTAV-IPN (los cuales pueden consultar e inscribirse a los distintos cursos que les permite el reglamento).* ***Coordinadores Académicos*** *quienes tienen todos los permisos para crear las bases de datos de alumnos y cursos y agregar/modificar su contenido*
    - ***¿Algún comentario extra?***
10. ***¿Debe el sistema trabajar bajo ciertas restricciones que no se han completado hasta el momento?***
11. ***¿Existen políticas internas que deban consultarse para que la información que genere el sistema sea válida para el CINVESTAV?***
12. ***Aproximadamente, ¿Qué tiempo dura el período de inscripciones? ¿Cómo considera que es, la mejor manera de manejar el período de inscripciones, para después de este lapso los alumnos no puedan realizar inscripciones?***

Los cuestionarios fueron aplicados a las siguientes personas usuarias finales del SIV o que interaccionan con el SIV.

Usuario entrevistado	Cargo
Dr. Pedro Mejía Álvarez	Contratista del SIV
Dr. Jorge Buenabad Chavez	Coordinador Académico de la sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN

Apéndice B

Cuestionarios

Aplicaciones de cuestionarios a usuarios del SIV

De los cuestionarios que fueron aplicados se obtuvo información para la identificación de los requerimientos tanto de usuarios como requerimientos del sistema. Las preguntas elaboradas en los cuestionarios fueron:

1. *Se desea particularmente que se construya una interfaz gráfica amigable y eficiente.*
   - ***¿Cómo calificaría amigable?***
   - ***¿Para usted que sería eficiente?***
2. *El sistema SIV debe permitir a un alumno elegir los cursos que desea tomar en el cuatrimestre de la sección de departamento de Ingeniería Eléctrica a la que pertenezca.*
   - ***¿Bajo que condiciones se pueden elegir materias?***
   - ***¿Hay algún mapa curricular que deba seguir?***
3. *Sólo podrá hacer uso del sistema SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares (que no estén dados de baja) en alguna sección del DIE.*
   - ***¿Cómo se sabe que alumnos están regulares?***
4. *Todos los accesos al sistema SIV deberán realizarse desde una interfaz gráfica accesible desde Internet.*
   - ***¿Alguna característica en específico?***
5. *El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materia.*
   - ***¿Su tutor académico no tendría que autorizarlo?***
   - ***¿Cuál es la política de su departamento?***
6. *Así mismo, el sistema permitirá la creación y modificación de bases de datos conteniendo información de los cursos y de los alumnos inscritos en cada sección del DIE del CINVESTAV-IPN.*
   - ***¿Creación y modificación de bases de datos?***

- ***¿Cuál es la idea detrás de esto?***

7. *El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.*
   - ***¿Quién será el encargado de dar estas claves?***
   - ***¿Cómo se notificará a cada alumno de su clave?***
   - ***¿Se permitirá que cada alumno pueda personalizar su clave?***
8. *Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrán acceder la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.*
   - ***¿Quién será el encargado de dar éstas claves?***
   - ***¿Cómo se le notificará su clave?***
   - ***¿Qué se planea hacer?***
9. *Se pueden considerar las siguientes vistas al sistema:* ***Público en general*** *(los cuales solo pueden consultar datos de los cursos).* ***Alumnos*** *del CINVESTAV-IPN (los cuales pueden consultar e inscribirse a los distintos cursos que les permite el reglamento).* ***Coordinadores Académicos*** *quienes tienen todos los permisos para crear las bases de datos de alumnos y cursos y agregar/modificar su contenido*
   - ***¿Algún comentario extra?***
10. ***¿Debe el sistema trabajar bajo ciertas restricciones que no se han completado hasta el momento?***
11. ***¿Existen políticas internas que deban consultarse para que la información que genere el sistema sea válida para el CINVESTAV?***
12. ***Aproximadamente, ¿Qué tiempo dura el período de inscripciones? ¿Cómo considera que es, la mejor manera de manejar el período de inscripciones, para después de este lapso los alumnos no puedan realizar inscripciones?***

Los cuestionarios fueron aplicados a las siguientes personas usuarias finales del SIV o que interaccionan con el SIV.

Usuario entrevistado	Cargo
Dr. Pedro Mejía Álvarez	Contratista del SIV
Dr. Jorge Buenabad Chavez	Coordinador Académico de la sección de Computación de Ingeniería Eléctrica del CINVESTAV-IPN

Apéndice C

Matriz de Dependencias

Matriz de dependencia entre requerimientos

ID	RF1	RF1.1	RF2	RF2.1	RF2.2	RF2.3	RF2.4	RF2.5	RF2.6	RF2.7	RF2.8	RF2.9	RF2.10	RF2.11	RF2.12	RF2.13	RF2.14	RF2.15	RF2.16
RF1		x																	
RF1.1	x																		
RF2	X			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF2.1			x																
RF2.2			x																
RF2.3			x																
RF2.4			x																
RF2.5			x																
RF2.6			x																
RF2.7			x																
RF2.8			x																
RF2.9			x																
RF2.10			x																
RF2.11			x																
RF2.12			x																
RF2.13			x																
RF2.14			x																
RF2.15			x																
RF2.16			x																
RF3	X																		
RF3.1																			
RF3.2																			
RF3.3																			
RF3.4																			
RF3.5																			
RF3.6																			
RF3.7																			
RF4	x																		
RF4.1																			
RF4.2																			
RF4.3																			
RF4.4																			
RF4.5																			
RF4.6																			
RF4.7																			
RF4.8																			
RF5																			
RF5.1																			

X = Los requerimientos son dependientes

I = Los requerimientos están intercalados

C = Los requerimientos están en conflicto

* = Una celda vacía indica que los requerimientos son independientes

Continuación de la Matriz de dependencia para los requerimientos del SIV:

ID	RF3	RF3.1	RF3.2	RF3.3	RF3.4	RF3.5	RF3.6	RF3.7	RF4	RF4.1	RF4.2	RF4.3	RF4.4	RF4.5	RF4.6	RF4.7	RF4.8	RF5	RF5.1
RF1																			
RF1.1																			
RF2																			
RF2.1																			
RF2.2																			
RF2.3																			
RF2.4																			
RF2.5																			
RF2.6																			
RF2.7																			
RF2.8																			
RF2.9																			
RF2.10																			
RF2.11																			
RF2.12																			
RF2.13																			
RF2.14																			
RF2.15																			
RF2.16																			
RF3		x	x	x	x	x	x	x											
RF3.1	x																		
RF3.2	x																		
RF3.3	x																		
RF3.4	x																		
RF3.5	x																		
RF3.6	x																		
RF3.7	x																		
RF4										x	x	x	x	x	x	x	x		
RF4.1									x										
RF4.2									x										
RF4.3									x										
RF4.4									x										
RF4.5									x										
RF4.6									x										
RF4.7									x										
RF4.8									x										
RF5																			x
RF5.1																		x	

X = Los requerimientos son dependientes

I = Los requerimientos están intercalados

C = Los requerimientos están en conflicto

* = Una celda vacía indica que los requerimientos son independientes

Apéndice D

Matriz de Análisis

Continuación de la Matriz de dependencia para los requerimientos del SIV:

ID	Entendible	Necesario	Consistente	No ambiguo	Completo	Verificable	Correcto	Real	Rastreable	Modificable
RF1	si	si	si	si	no	si	no	si	si	si
RF1.1	si	si	si	si	si	si	si	si	si	si
RF2	si	si	si	si	no	si	si	si	si	si
RF2.1	no	si	si	no	no	si	si	si	si	si
RF2.2	no	si	si	no	no	si	si	si	si	si
RF2.3	no	si	si	si	si	si	si	si	si	si
RF2.4	si	-	si	si	no	si	si	si	si	si
RF2.5	si	si	si	no	no	si	si	si	si	si
RF2.6	si	si	si	no	no	si	si	si	si	si
RF2.7	si	si	si	no	no	si	si	si	si	si
RF2.8	si	si	si	si	no	si	si	si	si	si
RF2.9	si	si	si	si	no	si	si	si	si	si
RF2.10	si	si	si	si	no	si	si	si	si	si
RF2.11	si	si	si	si	no	si	si	si	si	si
RF2.12	si	si	si	si	si	si	si	si	si	si
RF2.13	si	si	si	si	no	si	si	si	si	si
RF2.14	si	si	si	si	si	si	si	si	si	si
RF2.15	si	si	si	si	si	si	si	si	si	si
RF2.16	si	si	si	si	no	si	si	si	si	si
RF3	si	si	si	si	si	si	si	si	si	si
RF3.1	si	si	si	si	no	si	si	si	si	si
RF3.2	si	si	si	si	no	si	si	si	si	si
RF3.3	no	si	si	no	no	si	si	si	si	si
RF3.4	si	-	si	si	si	si	si	si	si	si
RF3.5	si	si	si	si	no	si	si	si	si	si
RF3.6	si	si	si	si	si	si	si	si	si	si
RF3.7	si	si	si	si	si	si	si	si	si	si
RF4	si	si	si	no	no	si	si	si	si	si
RF4.1	si	si	si	no	no	si	si	si	si	si
RF4.2	si	si	si	si	si	si	si	si	si	si
RF4.3	si	si	si	si	no	si	si	si	si	si
RF4.4	si	si	si	si	si	si	si	si	si	si
RF4.5	si	si	si	si	si	si	si	si	si	si
RF4.6	si	si	si	si	si	si	si	si	si	si
RF4.7	si	si	si	si	si	si	si	si	si	si
RF4.8	si	si	si	si	si	si	si	si	si	si
RF5	si	si	si	si	si	si	si	si	si	si
RF5.1	si	si	si	si	si	si	si	si	si	si

Bibliografía

[1] Shari Lawrence Pfleeger, Ingeniería de Software, teoría y práctica, Prentice Hall, 2002.

[2] Sommerville Ian, Software Engineering, Wiley, 1998.

[3] Pressman S. Roger, Ingeniería del software, un enfoque práctico, Cuarta Edición, McGraw Hill, 1998.

[4] Braude J. Eric, Ingeniería de Software, una perspectiva orientada a objetos, Alfaomega, 2003.

[5] Jacobson Ivar, Booch Grady, Rumbaugh James, El Proceso Unificado de Desarrollo de Software, Addison Wesley, 2000.

[6] Van Vliet Hans, Software Engineering, Principles and Practice, Secund Edition, Wiley, 2000.

[7] Stevens Perdita, Pooley Rob, Utilización de UML en Ingeniería de Software con Objetos y Componetes, Primera Edición, Addison Wesley, 2002.

[8] Maciaszek A. Leszek, Requirements Análysis and System Design, Developing Information Systems with UML, First Edition, Addison Wesley, 2001.

[9] Bruegge Bernd, Dutoit H. Allen, Ingeniería de Software Orientada a Objetos, primera edición, Prentice Hall, 2002.

[10] Kendall E. Kenenth, Kendall E. Julie, Análisis y Diseño de Sistemas, primera edición, Prentice Hall Hispanoamericana, 1991.

[11] Kotonya Gerald, Sommerville Ian, Requirements Engineering, processes and techniques, Wiley, 2000.

[12] Lauesen Soren, Software Requirements, Styles and Techniques, first edition, Eddison Wesley a Pearson Education Book, 2002.

[13] Craig L., UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos, Pearson, 1999.

[14] Macaulay L. A., Requirements Engineering, Springer-Verlag, 1996.

[15] Morris D., Tamm B., Concise Encyclopedia of Software Engineering, Pergamon Press, 1993.

[16] Checkland P., Scholes J., Soft System Methodology in Action, Wiley & Sons, 1990.

[17] Burg J. F. M., Linguistics Instruments in Requirements Engineering, Thesis Doctoral, Vrije Universiteit, IOS Press, 1997.

[18] Reubenstein H. B., Waters R. C., The Requirements Apprentice Automated Assistance for Requirements Acquisition, IEEE Transactions on Software Engineering, Vol. 17, No. 3, pp.226-240, 1991.

[19] Templeton J.F., The Focus Group: A strategic Guide to Organizing, Conducting and Analyzing the Focus Group Interview, Probus publishing Co., 1994.

[20] Viller S., Social Analysis for Software Engineering, Technical Report 1998, http://www.comp.lancs.ac.uk/computing/research/cseg/98_rep.html

[21] Burg J. F. M., Van de Riet R. P., Análisis Informal Requirements Specifications: A First step towards Conceptual Modelling, http://www.cs.vu.nl/jmfburg/publications.html

[22] Cybulski J. L., Patterns in Software Requirements Reuse, http://www.dis.unimelb.edu.au./stajj/jacob/

[23] IEEE Definición de Ingeniería de Software, http://www.ieee.org.com

[24] The explosion of the Ariene 5, http://www.ima.umn.edu~arnold/disasters/ariane.html

[25] ARIANE 5, Flight 501 Failure Report by the Inquiry Board,

http://java.sun.com/people/jag/Ariane5.html

[26] ARIANE 5, Flight 501 Failure, http://www.dcs.ed.ac.uk/home/pxs/Book/ariane5rep.html

[27] Trevor Jenkins, London Ambulance Fiasco, http://catless.ncl.ac.uk/Risks/13.88.html#subj1

[28] Brian Randell, London Ambulance Service, http://catless.ncl.ac.uk/Risks/13.88.html#subj1

[29] Case Study: The London Ambulance Service, http://www.dcs.napier.ac.uk/~michael/hci_ohps/hci447la.html

[30] London Ambulance Service Case Study, Gary Hubbard, Vishal Khatnani, Tim Kyle, Edward Pennie, Bruce Philp., http://members2.easyspace.com/penniee/co617las.doc.

[31]London Ambulance Service Computer Aided Despatch, http://www.scit.wlv.ac.uk/~cm1995/cbr/cases/case12/12.HTM

[32] Terca-25 Accidents: An Updated Version of the Original Accident Investigation Paper, Nancy Leveson, http://sunnyday.mit.edu/therac-25.html

[33] VODRTools, Ian Sommerville, http://www.comp.lancs.ac.uk/computing/resources/SE6/

[34] RequisitePro, Rational Co., http://www.rational.com

[35] Catalize Enterprise, SteelTrace Ltd, http://www.gnu.org/

[36] Integral Requisite Analizer, TCP Sistemas e Ingeniería, http://www.tcpsi.es