ANÁLISIS Y EVALUACIÓN DE TÉCNICAS Y HERRAMIENTAS DE LA INGENIERÍA DE REQUISITOS BASADO EN EL ESTUDIO DE DOCUMENTOS, SOFTWARE HEREDADO Y USO DE PROTOTIPOS

NATALIE ANDREA DURÁN TORRES
CARLOS ANDRÉS GARCÍA MARIÑO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

ANÁLISIS Y EVALUACIÓN DE TÉCNICAS Y HERRAMIENTAS DE LA INGENIERÍA DE REQUISITOS BASADO EN EL ESTUDIO DE DOCUMENTOS, SOFTWARE HEREDADO Y USO DE PROTOTIPOS

NATALIE ANDREA DURÁN TORRES
CARLOS ANDRÉS GARCÍA MARIÑO

Trabajo de grado para optar al título de
Ingeniero de Sistemas

Directores:
Ing. JAVIER MEDINA CRUZ
Ingeniero de Sistemas

MSc. FERNANDO ANTONIO ROJAS MORALES
Magíster en Ciencia de Computadores

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

“Es, pues, la fe la certeza de lo que se espera, la convicción de lo que no se ve”. Hebreos 11:1

Dedico este trabajo a DIOS, por ser mi Roca y mi Fortaleza, mi sustento y mí guía en todo momento, sin su ayuda, esto no hubiese sido posible.

A mi madre María Teresa Torres G. y a mi padre Edilberto Duran P. por apoyarme y creer en mí.
A mi hijo, Iván Santiago Cruz Durán por ser fuente de mi inspiración para terminar esta etapa y salir adelante.
A mi abuela Silvina (Q.E.P.D.), quien en vida siempre me bridó su amor, sus enseñanzas y apoyo

A mis amigos y a todas aquellas personas que estuvieron presentes con sus oraciones y buenos deseos.

Natalie Andrea Durán Torres

“Incluso un camino sinuoso, difícil, nos puede conducir a la meta si no lo abandonamos hasta el final”

Paulo Coelho

A toda mi familia, en particular a mi madre María Yolanda, mi padre Carlos Arturo y mi hermana Laucaryoly, que me apoyaron y estuvieron siempre dispuestos a acompañarme.

Dedicado especialmente a mi princesa que me da el mejor motivo para seguir adelante.

Carlos A. García M.

AGRADECIMIENTOS

A la Universidad Industrial de Santander y a la Escuela de Ingeniería de Sistemas por su formación profesional.

A todo el personal de la Escuela de Ingeniería de Sistemas e Informática, en especial a María Cecilia, quienes por su colaboración nos permiten conseguir nuestros objetivos.

A los profesores el Fernando Rojas Morales y el Javier Medina Cruz de quienes recibimos gran ayuda y comprensión en la elaboración de nuestro trabajo.

A nuestros calificadores, Ing. Enrique Torres e Ing. Sergio Rico, por apreciar nuestro trabajo y por todos sus consejos.

A nuestros compañeros y amigos que siempre nos alentaron y pusieron todo de sí para apoyarnos y que cumpliéramos este objetivo.

Y a todos esas personas, familiares, amigos y compañeros que intervinieron para lograr esa meta.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

GLOSARIO

Análisis sintáctico: es el análisis de las funciones sintácticas o relaciones de concordancia y jerarquía que guardan las palabras agrupándose entre sí en sintagmas, oraciones simples y compuestas de proposiciones.

Análisis Semántico: El análisis semántico es posterior al sintáctico y mucho más difícil de formalizar que éste. Se trata de determinar el tipo de los resultados intermedios, comprobar que los argumentos que tiene un operador pertenecen al conjunto de los operadores posibles, y si son compatibles entre sí.

Analista: es aquel individuo que realiza la reingeniería o ingeniería inversa a un sistema existente.

Ambigüedad: se da cuando una palabra, sintagma u oración es susceptible de dos o más significados o interpretaciones

Aplicación (Interfaz) basada en texto: se trata de una aplicación que acepta entradas del usuario solo mediante un teclado.

Árbol Sintáctico: Ayuda a concluir si el conjunto está bien formado y equivale a una estructura que es reconocida para el lenguaje.

Bibliometría: engloba el estudio de los aspectos cuantitativos de la producción, diseminación y uso de la información registrada, a cuyo efecto desarrolla modelos y medidas matemáticos, que sirven para hacer pronósticos y tomar decisiones en torno a tales procesos.

Categorización: este concepto en RI, refiere a la clasificación ya sea de términos o de documentos.

Dispositivo de entrada: es una pieza hardware del computador la cual permite al usuario interactuar con él. El concepto de dispositivos lógicos, agrupa los dispositivos de entrada en cinco clases: localizador (posición), elector (selección), valorador (números), teclado (cadenas) y elecciones (acciones).

Elicitación: Consiste en Adquirir toda la información relevante y necesaria para producir un modelo de los requisitos de un dominio del problema

Entrada de datos: es el acceso de un usuario al computador por los dispositivos de entrada y que no se interpretan por la aplicación, pero se trata como información para ser almacenada.

Esquemas Pre-conceptuales: Es una representación intermedia entre las especificaciones textuales en lenguaje natural ("modelos verbales") y los diferentes esquemas conceptuales que permiten el modelamiento de una pieza de software

Estilos de interacción: se definen como un conjunto de objetos de interface y sus técnicas asociadas, que proveen de una vista del comportamiento del usuario frente al sistema.

Extracción de la Información: disciplina dentro del procesamiento del lenguaje natural (PLN) que supone una revolución tecnológica en el ámbito de la recuperación de información y que pretende agilizar la obtención de la información útil por parte de los usuarios.

Gestos: son una secuencia de interacciones del usuario con el sistema que pueden ser reconocidas como pertenecientes al conjunto de símbolos que tienen significado para la aplicación.

Heurística: Capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines.

Informetría: abarca el estudio de los aspectos cuantitativos de la información, independientemente de la forma en que aparezca registrada y del modo en que se genere.

Ingeniería inversa: es el proceso de analizar un programa software para crear una representación de más alto nivel de abstracción que el código fuente.

Lenguaje Natural: es el que nos permite el designar las cosas actuales y razonar a cerca de ellas, fue desarrollado y organizado a partir de la experiencia humana y puede ser utilizado para analizar situaciones altamente complejas y razonar muy sutilmente.

Metodología: Es el modelo del proceso, el cual contiene técnicas y herramientas bien documentadas que lo soportan [44].

Objetos de interacción: son controles como botones, barras deslizantes o menús que pueden ser directamente manipulados por el usuario. Cada objeto de interacción tiene asociada una acción o atributo en el modelo de datos de la aplicación.

Pantalla: en el ambiente del proyecto CelLEST se define como una “unidad distintiva de presentación de información, disponible para los usuarios en un punto particular del proceso de interacción” [65].

Precisión: porción del material recuperado que es realmente relevante

Ranking: El ranking se define, generalmente, como un conjunto de elementos con un orden; a los cuales se les otorga un valor de 0 a 1, mostrándose en primer lugar los más próximos al 1.

Recall: porción de material que ha sido recuperado.

Recuperación de la Información (IR):es la ciencia de la búsqueda de información en documentos, búsqueda dentro de los mismos, búsqueda de metadatos que describan documentos, o también la búsqueda en bases de datos relacionales, ya sea a través de internet, intranet, para textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante.

Reingeniería: es el uso de técnicas de recuperación del diseño para entender el código de un software heredado y así poderlo modificar.

Relevancia: se considera relevante cuando el contenido del mismo posee alguna significación o importancia con motivo de la pregunta realizada por el usuario, es decir, con su necesidad de información.

Roles Semánticos: Los roles semánticos describen la relación semántica (no gramatical) que los argumentos tienen con respecto al predicado (normalmente un verbo). Otros términos utilizados para su denominación son: roles temáticos, casos semánticos, relaciones temáticas, argumentos semánticos, roles de los participantes.

Semántica del control de terminal: se presenta cuando la posición del cursor tiene significado para el sistema en aplicaciones basadas en texto.

Software heredado: es una aplicación que fue desarrollada y mantenida por un periodo de tiempo, típicamente los diseñadores y desarrolladores originales no están disponibles para realizar el mantenimiento del sistema. Por esto las especificaciones y documentación del software heredado están obsoletas, así que la única fuente de información acerca del sistema es el código en sí.

Técnica: Se trata de una serie de pasos bien documentados y que tienen reglas para su rendimiento y criterios para verificar que se han completado. Una técnica usualmente aplica para un solo proceso en un modelo de procesos. Algunas veces incluye notaciones y/o herramientas.

Usabilidad: del inglés (usability / user-friendliness) se puede tomar como sencillez de uso, refiriéndose a una interfaz de usuario.

Widgets: se trata de otro nombre para los objetos de interacción del usuario, específicamente usados para describir los objetos de interacción X Windows. [62, 63]

RESUMEN

TITULO: ANÁLISIS Y EVALUACIÓN DE TÉCNICAS Y HERRAMIENTAS DE LA INGENIERÍA DE REQUISITOS BASADO EN EL ESTUDIO DE DOCUMENTOS, SOFTWARE HEREDADO Y USO DE PROTOTIPOS[*]

AUTORES: NATALIE ANDREA DURÁN TORRES[**]
CARLOS ANDRÉS GARCÍA MARIÑO[**]

PALABRAS CLAVES: Ingeniería de requisitos, Análisis de documentos, Software heredado, prototipos, Ingeniería inversa.

CONTENIDO:

Este proyecto se desarrolló para dar una solución a la problemática encontrada acerca de la disponibilidad de los usuarios en la etapa de elicitación de requisitos sumado a la mala comunicación entre los stakeholders y los analistas, provocando que no se identifiquen ni validen correctamente los requisitos elevando el tiempo y costos.

Esta investigación procura dar unas técnicas basadas en el análisis de documentos, software heredado y uso de prototipos como referencia para la tarea de especificación de requisitos, en las que la interacción con el usuario sea solamente la mínima necesaria y que en cambio se haga más efectiva la información reunida, la recopilación de la información se llevó a cabo en tres fases principales, en las cuales se pretende dejar una base del estudio del estado del arte de estas técnicas, para estudios futuros en Ingeniería del Software:

La fase 1 consta de una consulta bibliográfica de las técnicas y herramientas de la ingeniería de requisitos basado en el estudio de documentos, para realizar un estado del arte; la fase 2 se compone de una consulta bibliográfica de las técnicas y herramientas de la ingeniería de requisitos basado en el análisis del software heredado y el uso de prototipos, el objetivo es establecer un estado del arte y una base bibliográfica de referencia en el tema para los futuros estudios a realizar: finalmente en la fase 3 se hizo la recopilación de la información obtenida para presentarla a la comunidad universitaria, como base para futuros proyectos.

[*]Trabajo de Grado – Modalidad Investigación
[**] Escuela de Ingeniería de Sistemas e Informática UIS. Director: Ing. Javier Medina Cruz. Codirector: Msc. Fernando Antonio Rojas Morales.

ABSTRACT

TITLE: ANALYSIS AND EVALUATION OF TECHNIQUES AND TOOLS FOR REQUIREMENTS ENGINEERING BASED ON ANALYSIS OF DOCUMENTS, LEGACY SOFTWARE AND USE OF PROTOTYPES[1]

AUTHORS: NATALIE ANDREA DURÁN TORRES[**]
CARLOS ANDRÉS GARCÍA MARIÑO[**]

KEYWORDS: Requirements engineering, analysis of documents, legacy software, prototyping, reverse engineering.

CONTENT:

This project was developed to provide a solution to the problem found on the availability of users in the requirements gathering stage coupled with poor communication between stakeholders and analysts, causing it not properly identify or validate the requirements by raising the time and costs.

This research attempts to provide some techniques based on analysis of documents, legacy software and use of prototypes as a reference for the task of requirements specification, in which user interaction is only the minimum necessary and instead will become more effective the information gathered, the data collection was carried out in three main phases in which they wish to leave a base of study of the state of the art of these techniques for future studies in software engineering:

Phase 1 consists of a bibliographic research for techniques and tools for requirements engineering based on the study of documents, for a state of the art. Phase 2 consists of a bibliographic research techniques and tools for requirements based on Legacy software, and use of prototypes, the objective is to establish a state of art and a bibliographic database of reference for future studies to be performed. Finally in phase 3 was collection of information obtained for submission to the university community as a basis for future project.

[1]Degree Project – Research Modality.
[**]Faculty of Physical-mechanical Engineering. School of System Engineering.
Director: Ing. Javier Medina Cruz. Codirector: Msc. Fernando Antonio Morales Rojas.

INTRODUCCIÓN

Actualmente se presenta una pobre determinación de requisitos de sistemas por el poco tiempo que tienen los usuarios para se les realice una adecuada elicitación de estos; La resistencia de los usuarios al cambio y la falta de interés en la fase de elicitación de requisitos son las conductas más frecuentes. Este comportamiento se debe al poco entendimiento de la importancia de la fase de especificación de requisitos, por la empresa como para el proyecto en sí.

Otro inconveniente es la aparición de requisitos inviables lo que puede resultar en retrasos en la entrega de los proyectos por el replanteamiento de requisitos a unos nuevos y factibles. La responsabilidad de este tipo de problemas recae sobre el equipo de proyecto el cual debe especificar muy bien desde el principio los alcances del nuevo sistema teniendo en cuenta los recursos de la empresa.

La negativa de los usuarios a participar durante la fase de elicitación de los requerimientos se entiende porque algunos métodos de especificación de requerimientos pueden exigir un nivel de abstracción de los usuarios, y si esto no se presenta conllevara a una mala comunicación entre los usuarios y el equipo de proyecto y una mala comprensión del objetivo general y los alcances del proyecto.

Este proyecto brinda unas técnicas de referencia para la tarea de especificación de requisitos, en las que la interacción con el usuario sea solamente la mínima necesaria y que en cambio se haga más efectiva la información reunida.

Las técnicas se basan en el análisis de documentos y en el estudio del software heredado encontrado en el entorno del proyecto de desarrollo.

Este trabajo consta de 5 secciones divididas en presentación del proyecto, metodología y plan de trabajo, marco teórico, análisis de documentos y software heredado y uso de prototipos.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Analizar y evaluar técnicas y herramientas de la ingeniería de requisitos, tales como el análisis de documentos, software heredado y uso de prototipos GUI en la etapa de levantamiento de requerimientos.

1.2 OBJETIVOS ESPECÍFICOS

- Evaluar el Estado del Arte de Ingeniería de Requisitos, referentes al análisis de documentos, software heredado y uso de prototipos, con el fin de brindar un fundamento para el desarrollo de la investigación.

- Evaluar las técnicas de captura de requerimientos, encontrados en la bibliografía y documentados en otros proyectos, con base en el análisis de documentos.

- Evaluar el uso del análisis de software heredado en la captación de requerimientos, encontrado en la bibliografía y documentados en otros proyectos.

- Integrar a la investigación el uso de prototipos y el proceso de análisis de interfaces gráficas para la validación de los requerimientos por parte de los usuarios.

- Presentar las conclusiones de la investigación realizada.

2. JUSTIFICACIÓN

Los escenarios de desarrollo de software han sido muy cambiantes en lo que tiene que ver con sus paradigmas y con las competencias tecnológicas de los usuarios esto obliga a los desarrolladores a que analicen diversas adaptaciones a su metodología de desarrollo.

Existen metodologías de desarrollo, que recomiendan el uso del UML como lenguaje de modelado, cuya interpretación sigue siendo compleja para los usuarios de los sistemas de información y administradores de procesos en diversos tipos de negocios.

Se requiere el análisis de nuevos métodos o técnicas de desarrollo de software, que pueda tener en cuenta los nuevos escenarios del negocio y que permita ayudar a los desarrolladores a la obtención de proyectos óptimos.

Como se plantea, en el proceso de descubrir el propósito del nuevo sistema, se necesita “identificar” ciertas fuentes y en la actualidad a la fuente que más se plantea recurrir es a los usuarios, clientes y/o consumidores y en general, los interesados en el desarrollo (stakeholders), pero estas fuentes esconden ciertos problemas para el proceso, como: los stakeholders no saben lo que realmente quieren en el sistema, o que no hay una comunicación eficaz entre el analista y los stakeholders, malinterpretándose la información compartida o incluso encontrándose conflictos [11] además, actualmente la disponibilidad de los interesados es limitada provocando que no se identifiquen ni validen correctamente los requisitos.

Pero además de los stakeholders, en algunos proyectos, se cuenta con elementos del entorno como son los documentos y sistemas ya existentes [13], que podrían

ser fuentes para el proceso de la ingeniería de requisitos. La identificación y estudio de técnicas y herramientas para la utilización de estas fuentes en la elicitación de requisitos, es el tema de este proyecto.

En este trabajo se presentan técnicas basadas en el estudio de documentos, software heredado y uso de prototipos, con más detalle, y que se aplican en el entorno en el que se encuentran muchas de las empresas de nuestra región, es decir, en medio de la migración a la Web, con sistemas heredados que requieren un análisis y con pocas oportunidades de conseguir disponibilidad de tiempo de los usuarios, clientes o stakeholders para las actividades de elicitación de requisitos.

Se pretende que nuevos profesionales y estudiantes de pregrado valoren los procesos de la Ingeniería del software como ámbito investigativo, que pueda proporcionar soluciones adecuadas y bienestar social, en nuestro medio regional, nacional e internacional.

3. METODOLOGÍA Y PLAN DE TRABAJO

3.1 METODOLOGÍA

El trabajo de investigación se enmarcó en un estudio descriptivo y de investigación tecnológica.

A nivel descriptivo se plasmó el estado del arte de las técnicas de Ingeniería de requisitos: Análisis de documentos, software heredado y prototipado evolutivo, que permite evaluar y comprobar la eficiencia de estas.

La investigación se basa en lo propuesto por Watts [1] para la resolución de problemas en ciencia y tecnología con presentación de resultados prácticos, a partir del cual se definió una serie de fases, sus correspondientes actividades y resultados.

3.1.1 Estrategias para la recolección de Información:

La información necesaria para esta investigación se recopiló de las siguientes fuentes:

- Consultas bibliográficas en bases de datos especializadas
    - ACM
    - IEEE (Grupos especializados)
- Consultas en Internet y en bibliotecas
    - UNIRED
    - IBM
    - TYNER BLAIN
    - RQNG

- Manuales y documentación en general de herramientas software relacionadas
    - CelLEST
    - MORPH

3.2 PLAN DE TRABAJO

El proceso consta de cuatro fases, para las cuales esta definidos un conjunto de actividades que, basadas en lo propuesto por Watts, permiten obtener un mejor conocimiento para la solución de la problemática expuesta.

3.2.1 Primera Fase: Análisis para definir el estado del arte de la técnica de análisis de documentos:

3.2.1.1 Actividades. Estudio bibliográfico de las metodologías y técnicas existentes, orientadas al análisis de documentos y sus formas de aplicación.

Estado del arte de las técnicas encontradas orientadas al análisis documentos

3.2.2 Segunda Fase: Análisis para definir el estado del arte del análisis de software heredado y uso de prototipos:

3.2.2.1 Actividades. Estudio bibliográfico de las técnicas existentes de especificación de requisitos, orientadas al análisis de software heredado y uso de prototipos y sus formas de aplicación.

Estado del arte de las técnicas encontradas orientadas al análisis de software heredado y prototipos.

3.2.3 Tercera Fase: Análisis para definir el estado del arte de las interfaces graficas de usuario:

3.2.3.1 Actividades. Estudio bibliográfico de los métodos de captura de requerimientos basado en la Interfaz Gráfica de Usuario.

Estado del arte de las técnicas encontradas basadas en la Interfaz Gráfica de Usuario.

3.2.4 Cuarta Fase: Análisis y socialización de los resultados de la investigación:

3.2.4.1 Actividades. Evaluaciones realizadas a las técnicas encontradas. Definición de las ventajas, alcances y oportunidades de las técnicas de captura de requerimientos basadas en el análisis de documentos, software heredado, interfaces gráficas de usuario y uso de prototipos.

4. MARCO TEÓRICO

4.1 INGENIERÍA DE REQUISITOS

Uno de los factores más importantes en el éxito de un desarrollo software es la especificación formal y correcta de las funcionalidades y alcances que debe tener el nuevo sistema, es decir, “que” va a hacer, estas especificaciones se deben establecer durante una etapa temprana del desarrollo, en la mayoría de los casos identificada como la etapa de análisis de requisitos del sistema.

La razón de que se deba especificar muy bien las funcionalidades del sistema desde un principio, es documentada en numerosas oportunidades [6, 7, 8, 9], en resumen, se relaciona con que los costos de corregir los errores disminuyen cuanto más temprano se encuentren los errores y se realice la corrección. Una gráfica que explica esta conducta es la siguiente:

Figura 1. Conducta de Costo Vs. Etapa de Desarrollo – Modificado de [10]

En cuanto a las causas de los errores, según diferentes estudios [9], se hallan la falta de interacción con el usuario y la definición de requisitos incompletos o cambiantes, entre otras razones. Estas causas son susceptibles de mejorar si se enfrentan en una etapa temprana.

Formalmente las especificaciones de funcionalidad y alcance, se realizan bajo una rama de la ingeniería del software, que se denomina Ingeniería de Requisitos, y que se define como “el proceso de desarrollar una especificación de software” o “el proceso de establecer que servicios son requeridos y que limites hay en la operación y desarrollo del sistema” [11], esto define a la Ingeniería de Requisitos (IR) como un proceso, lo que le confiere características como: etapas, un orden, así como objetivos y técnicas para alcanzarlos.

Otra definición indica que la ingeniería de requisitos (IR) es “el proceso de descubrir el propósito para el cual el sistema software se va a construir, identificando los interesados en el sistema (stakeholders) y sus necesidades, documentando estas en forma que sea cómodo su análisis, comunicación y posterior implementación” [12].

El concepto principal de la IR es indiscutiblemente el requisito, y su tarea es identificar, documentar y validar un conjunto de ellos para que sean la especificación del dominio del problema, pero primero hay que puntualizar en que es un requisito.

4.1.1 Requisito. Una definición de requisito es la que brinda la IEEE en el estándar 610.12 de 1990 [14], en donde:

Un requisito es

(1) Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

(2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
(3) Una representación documentada de una condición o capacidad documentada como las descritas en (1) y (2).

Esta definición implica que los requisitos deben responder a las necesidades de los usuarios teniendo en cuenta la realidad del entorno establecido y que su especificación debe realizarse lo más completa posible.

Algunos tipos de especificaciones son confundidas con requisitos, como las Reglas del Negocio [15], los Detalles de Diseño, de implementación o de pruebas, la información relativa a la Planificación del Proyecto, las necesidades del Proyecto, etc. [10], por lo que este proyecto se concentra en los requisitos expresados en dos modelos:

- El modelo de casos de uso
- El modelo de clases del dominio del problema

Siendo los casos de uso los que brindan una visión más clara del nuevo sistema [16] y son más habituales [17]; y el modelo de clases del dominio del problema una base sólida para el diseño de software orientado a objetos.

4.1.2 Clasificación de los Requisitos. Según el estándar IEEE 380, los requisitos se pueden clasificar en funcionales, no-funcionales o restricciones [18]. Pero también cada autor clasifica los requisitos según su enfoque, lo que hace más difusos los esfuerzos de investigación en este campo, encontramos también otras clasificaciones siendo aún, los requisitos funcionales y no funcionales, los tipos más comúnmente aceptados.

4.1.2.1 Requisitos funcionales. Una definición dada por el estándar internacional 610.12 de 1990 es: “Son requisitos que especifican una función que un sistema o componente de un sistema debe estar habilitado para realizar” [14], es decir, son las funciones o servicios que el sistema debe realizar y como se debe comportar para dar estas funcionalidades o servicios a los usuarios, también es posible que pueda especificar aquello que el sistema no hace.

4.1.2.2 Requisitos no funcionales. A diferencia de los funcionales estos “Son requisitos que no conciernen específicamente con la funcionalidad del sistema pero ponen restricciones en el producto a ser desarrollado” [19]. Los requisitos no funcionales se refieren a propiedades que el sistema debe tener y que debieran ser especificados de manera que puedan ser medibles y controlados.

Hay diferentes tipos de requisitos no funcionales, recopilados de varias fuentes como estándares, el principal es el estándar IEEE 830 [20], y artículos [18], en esta clasificación están los requisitos de:

Interfaz	Privacidad	Rendimiento
Recursos	Usabilidad	Portabilidad
Verificación	Fiabilidad	Calidad
Seguridad	Disponibilidad	Restricciones

Las restricciones imponen los límites que rigen el desarrollo y la implementación del sistema, son muchas veces definidas por el cliente y/o por el entorno, y generalmente, no son observables por el usuario durante la ejecución del sistema.

Unas restricciones son de tipo tecnológico, otras temporal o incluso factores humanos, como un ejemplo de restricción tecnológica esta que: “se dispone de un sistema operativo específico”. En ocasiones son asimilados como requisitos no

funcionales ya que algunas veces una restricción implica uno o varios requisitos no funcionales, por ejemplo “se debe asegurar la usabilidad en el sistema operativo X disponible”.

Algunos autores complementan la clasificación de requisitos con conceptos como requisitos de información, de seguridad, objetivos del sistema [17], requisitos de interfaz [10], pero que otros autores incluyen en la definición de requisitos no funcionales [18].

4.1.3 Casos de uso. Hay diferentes formas de especificar los requisitos, que son alternativas al lenguaje natural [11] entre estas:

- Lenguaje natural estructurado
- Lenguajes de descripción del diseño
- Notaciones gráficas
- Especificaciones matemáticas

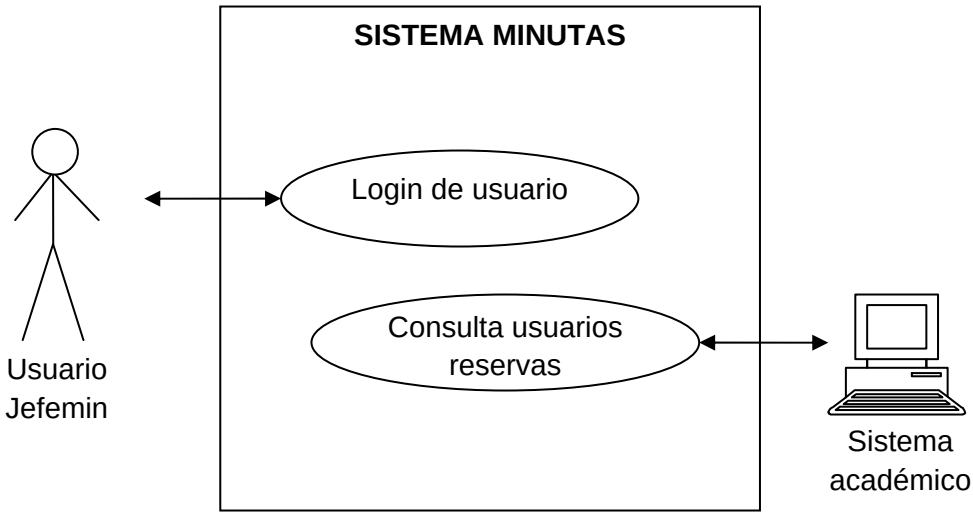
Las técnicas utilizadas como notación grafica están los diagramas de casos de uso que se plantean como una excelente fuente de realimentación y a la vez recolección de requisitos, que ayuda a la mejora continua del modelo de conocimiento del producto requerido [51]. La principal observación en el uso de los modelos visuales es que originan discusiones entre los stakeholders, que derivan en la aceptación o rechazo de los requisitos ya identificados y en la generación de nuevos requisitos.

Los diagramas de casos de uso pertenecen a las especificaciones gráficas, aunque aparte de la notación en diagramas también se pueden presentar los casos de uso de manera textual, en tablas o combinadas, cada una de las cuales tienen sus propias plantillas [20, 22, 23].

La técnica de los casos de uso se utiliza como una forma de especificar las funcionalidades externas del sistema y la relación con los actores. Una definición es “Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios” [24]. Aquí se diferencian dos elementos, por una parte el sistema, que es aquello que presta los servicios y alguien o algo que usa estos servicios, definido como un actor. Entonces dados estos elementos, es necesario representar sus interacciones, que en últimas, representaran la forma de cómo se comporta el sistema.

Un ejemplo de la notación gráfica se muestra a continuación:

Figura 2. Notación de Casos de Uso – Elaboración propia



Acá se puede apreciar, el sistema, encerrado en una caja que lo delimita, y los actores, elementos externos al sistema y cuyo tipo puede variar desde un usuario hasta otros sistemas. La representación específica de los casos de uso es que son

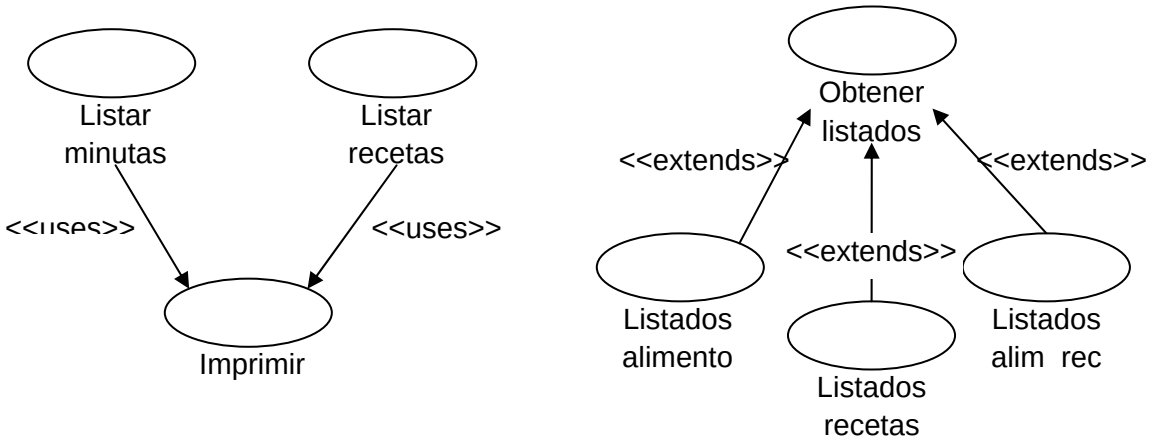
óvalos nombrados dentro del sistema, que se relacionan con actores y otros casos de uso.

La relación de un actor a un caso de uso se representa por una flecha simple que va desde el actor al caso de uso que invoca, es la relación más sencilla. Las relaciones de los casos de uso con otros casos de uso, están definidas también y tienen sus propias notaciones.

Las siguientes relaciones se pueden clasificar dentro de las relaciones de generalización. La relación de uso o inclusión, plantea que un Caso de Uso implica la ocurrencia de otro como parte de su procesamiento normal, esta relación también representa el uso, en donde un(os) caso(s) de uso, utiliza otro, la relación actualmente se denomina <<include>> que reemplazo al <<uses>>.

La relación de extensión, se presenta cuando un Caso de Uso implica la ocurrencia de uno u otro(s) casos de uso a fin de presentar diferentes opciones o formas de realizar su proceso, este tipo se define como <<extends>>. La figura 3 muestra ejemplos de las diferentes relaciones.

Figura 3. Ej. Relación de uso y extensión o herencia entre casos de uso – Elaboración propia

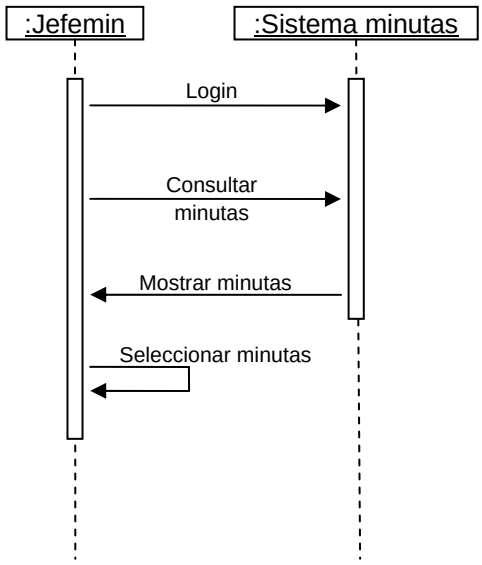


En las relaciones de generalización también se puede presentar un carácter de herencia en el que el Caso de Uso origen hereda la especificación del Caso de Uso destino y posiblemente la modifica y/o amplía.

Hay una noción importante respecto a los casos de uso y es la granularidad, que define el nivel de especificación de las funciones que se representan, en nuestro proyecto se define la granularidad, también llamado nivel de abstracción, de acuerdo a la interfaz gráfica de usuario, lo que a nuestro juicio nos lleva a pensar en que este nivel de abstracción valga como un mejor medio de comunicación para con el usuario, y así mismo como los prototipos que se realicen basados en estos.

Los escenarios son descripciones formales del flujo de eventos que ocurren durante una instancia de un Caso de Uso y pueden corresponder a una representación del diagrama de secuencia [25], este diagrama es la representación donde se puede ver más claramente un escenario como el comportamiento de la colaboración entre actores [26], un ejemplo se muestra a continuación:

Figura 4. Diagrama de secuencia – Elaboración propia



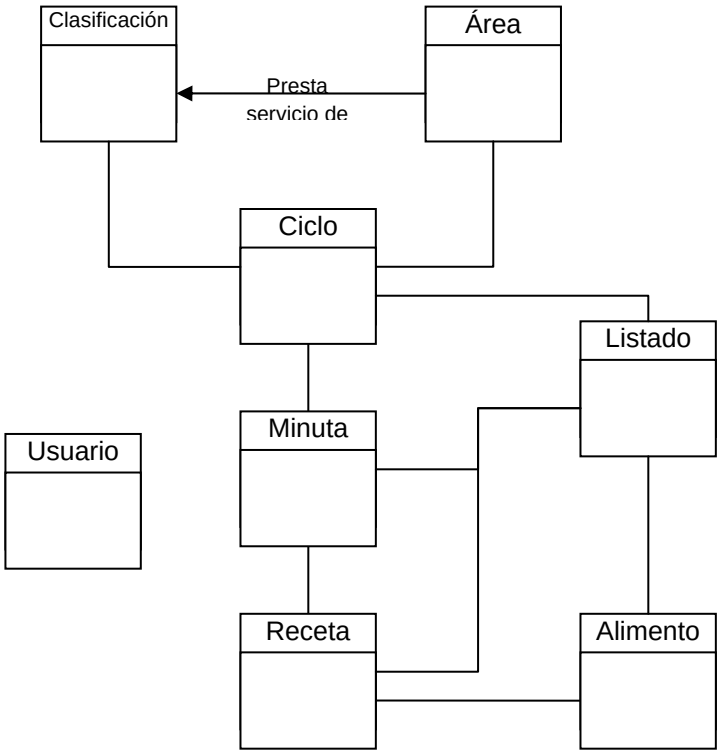
Para el proyecto la notación en diagramas se complementara con los detalles de la especificación textual de un caso de uso, aprovechando las plantillas establecidas en [22] y las descripciones en [24]. Las técnicas para extraer la información y desarrollar el caso de uso, se muestran en un apartado posterior.

4.1.4 Diagramas de clases del dominio del problema. Para la especificación del modelo de requisitos, según [27] se usan también las vistas de las clases participantes (VOPC - View of Participating Classes), en donde se muestran los actores y entidades que participan en un caso de uso específico. Estas vistas o diagramas hacen de especificación del modelo conceptual del dominio, y son basados en el diagrama de secuencia y en el diagrama de procesos. Ya se ha visto un ejemplo de un diagrama de secuencia pero un diagrama de procesos tiene un poco más de información ya que combina el flujo de trabajo con los datos y actividades para lograr un objetivo, mostrando lo que los actores realizan durante el flujo. La extracción de clases es posible a partir de los objetos de información que se presentan en estos diagramas [26]. Un ejemplo de diagrama de clases se muestra en la figura 5.

La representación de las clases en UML consiste en un nombre para la entidad, los atributos y los métodos pertenecientes a esta, además del nombre y cardinalidad de las respectivas relaciones entre cada entidad.

En este proyecto se buscaran las técnicas que den un primer acercamiento al modelo de clases del dominio de un problema, ya que solo aparecerán aquellas clases que se puedan extraer del diagrama de procesos, siendo estas las más visibles desde el punto de vista de la ejecución del sistema.

Figura 5. Ejemplo del Diagrama de clases del dominio del problema – Elaboración propia



4.1.5 El Proceso de la Ingeniería De Requisitos. Los diferentes autores [7], [10], [18], [46], fragmentan el proceso de ingeniería de requisitos en varios esfuerzos, entre los principales está el desarrollo y la gestión de los requisitos. En el desarrollo se encuentran tareas como la especificación (captura), especificación (modelado), análisis, documentación y validación, cuya finalidad es identificar y documentar los requisitos que se van hallando, cuidando de que se cumplan las características deseables en los requisitos (Posibles, Necesarios, Priorizados, Concretos y Verificables) [10] y en su especificación (Correcta, Completa, Consistente, Comprobable, Modificable, etc.) [20].

La gestión de los requisitos se centra en la administración de los cambios que se originan durante todo el proceso. Proveyendo de técnicas para el seguimiento de los cambios y el control de las versiones de la especificación entre otros.

Tabla 1. Proceso de la ingeniería de requisitos – Elaboración propia

Proceso principal	Sub-Proceso
Desarrollo	Elicitación
	Especificación
	Análisis
	Documentación
	Validación
Gestión	Control de cambios
	Re-documentación

Este proyecto se concentra en la tarea de elicitación de requisitos basada en el análisis de documentos y de software heredado, su integración con las otras tareas (proveyendo los documentos base para el análisis, especificación y documentación de requisitos), las técnicas de captura y algunas herramientas disponibles. Además analiza el uso de los prototipos, basados en la interfaz de usuario, para la validación de los requisitos.

4.1.6 La Elicitación De Requisitos. Esta tarea consiste en adquirir toda la información relevante y necesaria para producir un modelo de los requisitos de un dominio de problema. Se soporta en una serie de técnicas que realizan la toma de requisitos en los diferentes entornos que se encuentren y de las diferentes fuentes que estén disponibles.

Según el Manual de la Ingeniería de Requisitos [6] y el artículo web Diez Técnicas de Elicitación de Requisitos [28] hay una lista de las técnicas de elicitación más efectivas, entre las cuales se mencionan:

- Casos de uso
- Análisis de documentos
- Prototipado
- Storyboards
- Escenarios
- Ingeniería inversa [29, 30, IEEE]
- Análisis de interfaces

Ya que para esta tarea existen diversas técnicas, en la práctica se seleccionan de acuerdo a varias razones. En el artículo Caracterización de las Técnicas de Adquisición de Requisitos [31] se establecen algunos criterios usados por

diferentes autores pero se determina que no hay suficientes publicaciones que establezcan las características de las técnicas de manera científica y que estipulen como escogerlas según el entorno.

Según los autores de este trabajo, las técnicas se eligen dependiendo del tiempo y los recursos con los que cuente el analista y de la información que se espera obtener. Siendo estas características dependientes del tipo de proyecto que se va a realizar, por ejemplo, el tiempo puede ser limitado, como en el caso de los mantenimientos (se requiere que estén listos rápido para que no haya perdidas por parálisis en el funcionamiento de la empresa) o puede llevar más tiempo, como en los esfuerzos de evolución que contemplan la reingeniería del sistema total o parcialmente, o por ultimo en el caso del desarrollo de una aplicación completamente nueva en el cual se debe tomar todo el tiempo necesario para cada etapa del proceso.

También, los recursos o fuentes con que cuenta el analista son diferentes en el caso de una aplicación innovadora, a cuando se tiene un entorno con aplicaciones ya existentes o software heredado. En el primer caso se da más recurrir a documentos y expertos en el dominio como fuentes de información para la definición de requisitos, y en un entorno con aplicaciones heredadas estas aplicaciones son una fuente de información sobre el comportamiento y funcionalidad de la empresa. En este sentido y aunque el software heredado se defina en el articulo comprensión y evolución del software heredado [13] como: “sistemas que se implementaron años antes, su tecnología se convirtió en obsoleta, su estructura está deteriorada, contiene reglas del negocio que no se aplican en toda la organización, no se puede reemplazar fácilmente, y los autores no están disponibles” es preferible su evolución a su desatención.

Por las características del entorno se eligen técnicas que por un lado, resuelven la cuestión de la falta de tiempo de los usuarios o clientes para dedicarle a la etapa

de elicitación y por otro lado recopilan la suficiente información para dar un buen soporte a las siguientes etapas del proyecto. Estas técnicas son el análisis de documentos, el análisis del software heredado y el uso de prototipos.

Por otra parte los procesos más frecuentes para el software heredado son los de mantenimiento, evolución, reingeniería, y parte de los esfuerzos de cada uno de estos procesos, se centra en la comprensión del sistema, una técnica de la reingeniería, dirigida a esta tarea, es la ingeniería inversa.

5. ANÁLISIS DE DOCUMENTOS

5.1 EVOLUCIÓN HISTÓRICA

La sociedad ha ido evolucionando a partir de del perfeccionamiento de la información, lo que ha creado diversidad de técnicas y metodologías como una necesidad para su correcto manejo. Las primeras formas de la información fueron representadas en piedras y jeroglíficos (pictogramas e ideogramas), tablas, etc. Así, desde la antigüedad, se fueron conformando las bibliotecas como forma de preservación del conocimiento, siendo el libro escrito la representación más común; sin embargo, en la actualidad esta ha sido reemplazada, aunque no en su totalidad, por versiones digitales.

5.2 RECOPILACIÓN DE INFORMACIÓN

Dentro de los métodos para recopilar información, se encuentra el Análisis de Documentos, el cual puede usarse como base en la elicitación de requisitos a través de técnicas apoyadas en la Recuperación de la Información y en el Lenguaje Natural.

5.2.1 Análisis de Documentos. El análisis de documentos implica el reconocimiento y extracción de las partes textuales de los mismos y su pre-procesado (segmentación léxica y oracional, análisis y desambigüación morfosintáctica, detección y clasificación de entidades, análisis sintáctico, superficial y profundo, análisis semántico, resolución de correferencias, etc.).

5.2.1.1 Análisis de Documentos Cuantitativos. Se dispone de una variedad de documentos cuantitativos para su interpretación en cualquier negocio. Estos

incluyen reportes usados para la toma de decisiones, reportes de desempeño y diversas formas. Todos estos documentos tienen un objetivo específico y una audiencia hacia la que están dirigidos.

- Reportes usados para la toma de decisiones.
- Reportes de desempeño.
- Registros.
- Formas para captura de datos.

5.2.1.2 Análisis de Documentos Cualitativos. Muchos documentos que circulan dentro de las organizaciones no son cuantitativos. Aunque los documentos cualitativos tal vez no sigan una forma predeterminada, el análisis de ellos es crítico para la comprensión de la manera en que los miembros de la organización engranan en el proceso o en la organización.

Los documentos cualitativos incluyen:

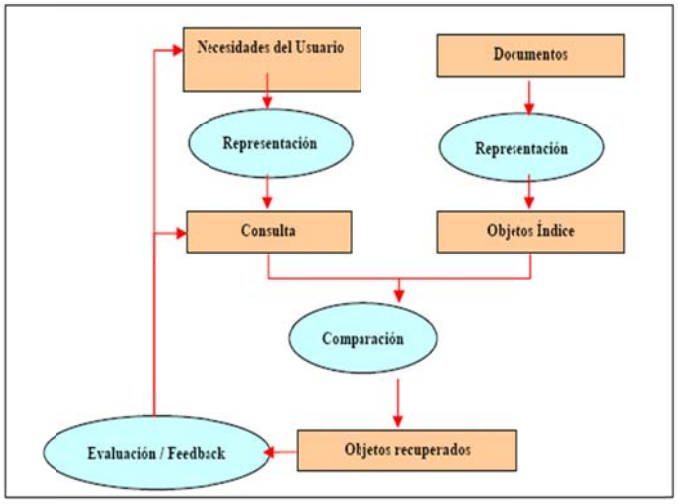
- Memorándums
- Consignas en un tablero o en áreas de trabajo
- Manuales de procedimientos
- Manuales de política
- Páginas WEB y correos electrónicos

5.3 RECUPERACIÓN DE LA INFORMACIÓN

5.3.1 La Recuperación. Según Croft, se trata de las tareas mediante las cuales se puede localizar y acceder a los recursos de información que son pertinentes para la resolución del problema planteado.

Los pasos se pueden ilustrar en la Figura 6. Comenzando desde la identificación de las necesidades del usuario y la consulta a las fuentes de información hasta el procesamiento, evaluación y selección de resultados.

Figura 6. Proceso de Recuperación de la Información – Tomado de [88].



Teniendo en cuenta lo anterior, se puede decir que la Recuperación de la Información tiene la tarea de organizar documentos (texto, multimedia, etc.), pertenecientes a una colección con una relevancia estimada para las necesidades de información de un usuario, que son previamente expresadas por éste último en función de las respuestas obtenidas a un requerimiento en un lenguaje no formal. Actualmente la RI (Recuperación de la Información), como ejemplo Google, AltaVista, etc. Llenan las necesidades de los usuarios de forma parcial, debido al gran número de información irrelevante que recuperan hace que no se aprovechen de manera significativa los recursos de la web.

5.3.2 Clasificación de documentos. Clasificación no Supervisada

Tabla 2. Clasificación de Documentos. Elaboración Propia

	Clasificación supervisada
En este escenario no hay categorías previas ni esquemas o cuadros de clasificación establecidos a priori. Los documentos se agrupan en función de ellos mismos, de su contenido; de alguna manera, podemos decir que se auto organizan	Se presenta en, una situación en la que se parte de una serie de clases o categorías conceptuales prediseñadas a priori, y en la que labor del clasificador (manual o automático) es asignar cada documento a la clase o categoría que le corresponda.

5.3.2.1 Algoritmos Aplicados a la Clasificación de Documentos:

5.3.2.1.1 Algoritmos probabilísticos. Se basan en la teoría probabilística, en especial en el teorema de Bayes [89]. Se trata de estimar la probabilidad de que un documento o un término pertenezcan a una categoría, esto depende de la posesión de una serie de características, de cada una de las cuales conocemos la probabilidad de que aparezcan en los documentos que pertenecen a la categoría en cuestión.

Con dichas probabilidades obtenidas de la colección de entrenamiento, podemos estimar la probabilidad de que un nuevo documento, dado que contiene un conjunto determinado de términos, pertenezca a cada una de las categorías.

5.3.2.1.2 Algoritmo de Rocchio. Formulada y ejecutada una primera consulta, el usuario examina los documentos devueltos y determina cuáles le resultan relevantes y cuáles no. Con estos datos, el sistema genera automáticamente una nueva consulta, basándose en los documentos que el usuario señaló como relevantes o no relevantes. Por tanto, el algoritmo de Rocchio proporciona un

sistema para construir el vector de la nueva consulta, re-calculando los pesos de los términos de ésta y aplicando un coeficiente a los pesos de los la consulta inicial, otro a los de los documentos relevantes y otro distinto a los de los no relevantes.

Una vez que se tienen los patrones de cada una de las clases, el proceso de entrenamiento o aprendizaje está concluido. Para categorizar nuevos documentos, simplemente se estima la similitud entre el nuevo documento y cada uno de los patrones. El que arroja un índice mayor nos indica la categoría a la que se debe asignar ese documento.

5.3.2.1.3 Algoritmo del vecino más próximo. El algoritmo de basa en localizar el documento más similar o parecido al que se desea clasificar. Para esto no hay más que utilizar ese documento como si fuera una consulta sobre la colección de entrenamiento. Una vez localizado el documento de entrenamiento más similar, dado que éstos han sido previamente categorizados manualmente, sabemos a qué categoría pertenece y, por ende, a qué categoría debemos asignar el documento que estamos clasificando.

5.3.2.1.4 Redes neuronales. Una de las principales aplicaciones de las redes neuronales es el reconocimiento de patrones. Básicamente, una red neuronal consta de varias capas de unidades de procesamiento o neuronas interconectadas; en el ámbito que nos ocupa la capa de entrada recibe términos, mientras que las unidades o neuronas de la capa de salida mapea clases o categorías. Las interconexiones tienen pesos, es decir, un coeficiente que expresa la mayor o menor fuerza de la conexión.

Es posible entrenar una red para que, dada una entrada determinada (los términos de un documento), produzca la salida deseada (la clase que corresponde a ese

documento). El proceso de entrenamiento consta de un ajuste de los pesos de las interconexiones, a fin de que la salida sea la deseada.

5.3.3 Técnica de RI para la elicitación de requisitos a partir de código fuente usando Modelos de Espacios Vectoriales. En este enfoque, hay dos partes principales:

- Recuperación de la información (aplicando modelo de espacio vectorial para indexar documentos, consultas y clasificación de resultados)
- Análisis estático de la estructura del código fuente

El modelo de espacio vectorial [90] [91] propone un marco en el que la coincidencia parcial es posible. Trata de consultas y documentos como vectores construidos por términos índice. Estos términos se adquieren en los textos de consultas y documentos de acuerdo a unas reglas (haciendo caso omiso a artículos, puntuaciones, números, etc). Cada término índice tiene un peso distinto en diferentes documentos y vectores de consulta.

Vector ($w_{1,q,}$ $w_{2,q}$, ... ,$w_{t,q}$) representa la consulta q, en la que $w_{i,q}$ es peso en el *q-ésimo* término índice en la consulta q y t es el número de términos índice

Vector ($w_{1,j}$, $w_{2,j}$, ... ,$w_{t,j}$) representa el documento d_j, en la que w_{ij}es el peso del *j-ésimo* término índice en el documento d_jy t es el número de términos índice.

El modelo de espacios vectoriales propone evaluar el grado de similitud de un documento d_j con respecto a la consulta q como correlación.

Esta correlación puede ser cuantificada por el coseno del ángulo entre estos dos vectores.

$$sim(d_j, q) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} Wi,j \times Wi,q}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

La idea básica de este enfoque es combinar métodos de RI con el análisis de la estructura del programa para localizar una función establecida según determinados requisitos.

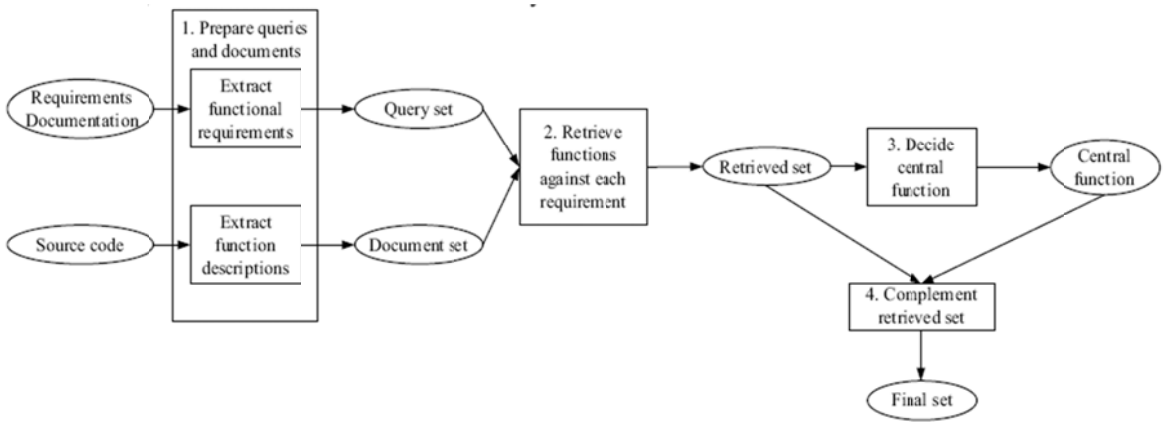
Debido a que un requisito contiene algunos datos de la implementación, se puede usar la tecnología de Recuperación de la Información para recuperar funciones que se relacionan con la información de la implementación. Estas funciones pueden ser asumidas como aquellas que se invocan al ejecutar un requerimiento. Por otra parte el gráfico de llamadas extraído del código fuente se utiliza para complementar los resultados obtenidos a través de RI.

En la RI, se usa cada requisito funcional adquirido en la documentación del sistema de software como consulta para recuperar las funciones de la misma. Así, la descripción de cada función es tratada como un documento en el proceso de recuperación de la información.

Para obtener la descripción de cada función, se extrae el nombre de la función, los nombres de los parámetros, los nombres de las variables locales, y los nombres de todas las funciones invocadas directamente en la función. Para complementar esto, en primer lugar se debe identificar la función las importante (Función central) en el conjunto recuperado. Luego añadimos funciones en algunas rutas relacionadas al último conjunto de recuperación. El gráfico de llamados en el

código fuente se puede utilizar para encontrar esas rutas. La estructura del enfoque se puede ver en la Figura 7.

Figura 7. Enfoque del proceso. Tomado de [92]



El proceso de este enfoque está descrito en la figura. Esta muestra 4 pasos principales:

(1) Preparar las consultas y los documentos a recuperar. Como se mencionó antes, la principal tarea de este paso es obtener el conjunto de consultas (el conjunto de requisitos) de la documentación de requerimientos y del conjunto de documentos (conjunto de descripción de funciones) de identificadores extraído del código fuente.

(2) Coincidencia de cada requerimiento con su respectiva función. Así para cada requisito podemos obtener una lista de funciones calificadas por la similitud con el requisito.

(3) En este paso se identifica la función central para recuperar cada requisito. El cálculo de la importancia de cada función se basa en la similitud de los valores adquiridos en la etapa de recuperación.

(4) Complementar la función de recuperación de conjuntos utilizando el gráfico de llamados extraído.

5.3.3.1 Preparando Consultas y Documentos. El conjunto de consulta se forma de la siguiente manera: de la documentación de requisitos, se selecciona un párrafo de texto para cada requisito funcional. Por lo general, todos los párrafos están en lenguaje natural. A continuación cada párrafo se transforma en un conjunto de términos índice según las prácticas habituales de RI, es decir, sólo los sustantivos y los verbos en el párrafo se consideran en la transformación, y éstas se normalizarán a su forma original (es decir, la única forma de nombres y el infinitivo de los verbos) para ser los términos índice finales. Cada conjunto de términos del índice es una consulta dentro del conjunto de consultas.

El conjunto de documentos se obtiene a partir del código fuente de la siguiente manera. Para cada función en el código fuente se extrae el conjunto de identificadores asociados a la función, estos identificadores deben incluir el nombre de la función, los nombres de los parámetros de la función, los nombres de las variables locales en la función, y los nombres de todas las funciones invocadas directamente en la función.

5.3.3.2 Recuperando el conjunto inicial de funciones. Luego de tener tanto el conjunto de consultas como el conjunto de documentos, se usa el modelo de espacios vectoriales para la recuperación.

Por cada consulta en el conjunto de consultas, se va a recuperar un subconjunto de documentos del conjunto de documentos ya calificados por su similitud entre la consulta y cada documento en el subconjunto.

Para este proceso se tiene en cuenta el modelo de espacios vectoriales [90], [91].

5.3.3.3 Identificando la función Central en el conjunto de Funciones Recuperadas. La identificación de la función central de un conjunto de funciones obtenidas en cada conjunto recuperado (paso 2 de la figura 11) presenta problemas debido a que algunas de las funciones relacionadas podría no tener ningún identificador. Sin embargo este enfoque busca una solución haciendo un llamado a las relaciones entre las funciones para poder obtener las funciones perdidas. Esto también trae dificultades ya que ésta podría introducir funciones no relacionadas.

5.3.3.4 Complementando la recuperación de los conjuntos de funciones. Luego de tener identificada la función central, se comienza el proceso de complementación. La idea principal es complementar el conjunto de funciones relacionadas tanto con la función central y con las no centrales usando diferentes estrategias. El último conjunto se complementa con base en el conjunto inicial.
El proceso de la complementación toma tres pasos principales:

i) Visitar todos los caminos que van desde la función de entrada a la función central, y añadir las funciones de estas rutas de acceso a la función de conjunto final.(Figura 8)

ii) Visitar todos los caminos que van de la función central a todos los nodos añadiendo las funciones en estas rutas de acceso a la función de conjunto final.

iii) Para cada una de las funciones no centrales en el conjunto inicial, salvo los de los caminos tanto de la función de entrada a la función central y de la función central a todos sus nodos, visita todos los caminos de ella en forma descendiente y se añaden las funciones de estas rutas para el conjunto final de la función.(Figura 9)

Figura 8. Recuperación inicial de funciones. Tomado de [92]

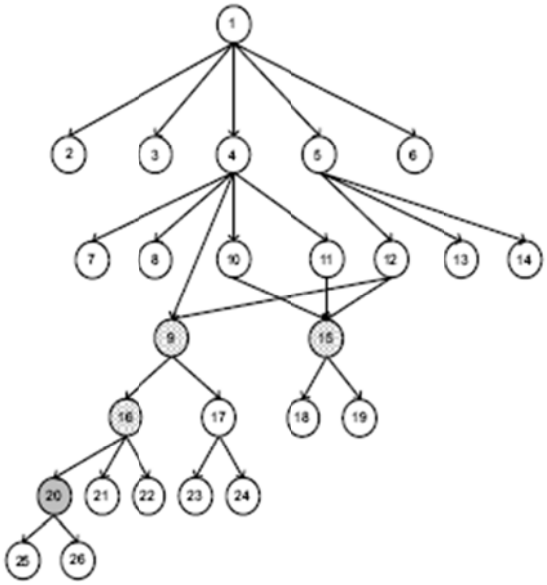
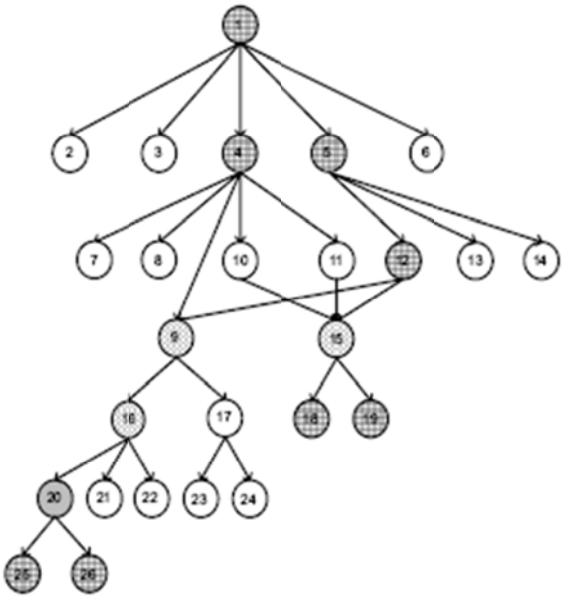


Figura 9. Conjunto final de funciones después de la complementación. Tomado de [92]



5.3.3.5 Proyecto SMART. Es una herramienta para la recuperación de la información, que implementa los modelos de espacios vectoriales propuestos por Salton en los 60's. El primer propósito de SMART es proveer un marco en el que se lleva a cabo una investigación en la recuperación de la Información, por lo tanto esta herramienta se puede ver como una versión estándar de indexación, recuperación y evaluación para el modelo del espacio vectorial.

Teniendo en cuenta lo visto antes, para cada requisito se selecciona la función central y se complementa para decidir la función necesaria para la exigencia.

5.3.3.5.1 Resultados. Después de probar la herramienta, se obtienen tres grupos:

(1) Conjunto de funciones precisas que se relacionan con los requisitos funcionales, que son adquiridos a través de la comprensión del código fuente por ingenieros experimentados.

(2) Recuperación de conjuntos de funciones, cada una de ellas relacionada con un requisito funcional.

(3) Conjunto final de funciones ya complementados. El primer grupo es usado para la comparación de conjuntos, se comparan el resto de los grupos usando dos métricas, "precisión" y "recall", que son usados en la práctica de IR.

Los resultados mostraron que para recuperación de requisitos funcionales usando IR a partir del código fuente, la precisión se incrementó en un14,228% y el recall en un 42,422%, lo que demuestra que esta herramienta resulta efectiva.

5.4 PROCESAMIENTO DEL LENGUAJE NATURAL

5.4.1 Lenguaje Natural. Medio que se utiliza comúnmente para establecer comunicación entre las personas.

Este tipo de lenguaje es el que nos permite el designar las cosas actuales y razonar a cerca de ellas, fue desarrollado y organizado a partir de la experiencia humana y puede ser utilizado para analizar situaciones altamente complejas y razonar muy sutilmente. La riqueza de sus componentes semánticos da a los lenguajes naturales su gran poder expresivo y su valor como una herramienta para razonamiento sutil. Por otro lado la sintaxis de un LN puede ser modelada por un lenguaje formal, similar a los utilizados en las matemáticas y la lógica. Otra propiedad de los lenguajes naturales es la poli-semántica, es decir la posibilidad de que una palabra en una oración tenga diversos significados.

5.4.2 Procesamiento del Lenguaje Natural (PLN). EL PLN es la utilización de un lenguaje natural para la comunicación con la computadora, mediante el entendimiento de las oraciones que le son proporcionadas.

El uso de estos lenguajes facilita el desarrollo de programas que realizan tareas o modelos relacionados con el lenguaje y que ayudan a comprender los mecanismos humanos relacionados con el lenguaje.

EL uso del lenguaje natural en la comunicación hombre-máquina tiene a su vez una gran ventaja, ya que su interlocutor no tiene que aprender el medio de comunicación, sin embargo su uso es también un obstáculo, porque la computadora tiene una limitada comprensión del lenguaje. Por ejemplo, el usuario no puede hablar con sobrentendidos, ni introducir nuevas palabras, ni construir sentidos derivados, que son tareas que se realizan espontáneamente cuando se utiliza en el lenguaje natural.

5.4.3 Arquitectura de un Sistema de Procesamiento de Lenguaje Natural (PLN). Se refiere a cómo la computadora va a analizar e interpretar las oraciones que le sean proporcionadas.

a) El usuario le expresa a la computadora que es lo que desea hacer.

b) La computadora analiza las oraciones proporcionadas en el sentido morfológico y sintáctico, es decir, si las frases contienen palabras compuestas por morfemas y si la estructura de las oraciones es correcta.

c) El siguiente paso es analizar las oraciones semánticamente, es decir, saber cuál es el significado de cada oración y asignar el significado de éstas a expresiones lógicas (cierto o falso)

d) Una vez analizadas las oraciones, de analizan todas juntas, tomando en cuenta la situación de cada oración respecto a las oraciones anteriores, así la computadora ya sabe que es lo que va a hacer, es decir, ya tiene la expresión final.

e) Ya teniendo la expresión final, el siguiente paso es la ejecución, para obtener así el resultado y proporcionarlo a usuario.

5.4.4 Ambigüedad en el Lenguaje Natural. La ambigüedad es un proceso lingüístico que se presenta cuando pueden admitirse distintas interpretaciones a partir de una representación dada. Para desambiguar, es decir, para seleccionar los significados más adecuados de un conjunto conocido de posibilidades, existen diversas estrategias de resolución según el caso [93].

La ambigüedad es el problema más importante en el procesamiento de textos en LN, por lo que resolver este problema es la tarea más importante.

5.4.5. Técnica para obtener esquemas pre-conceptuales para la especificación de requerimientos a partir documentos en Lenguaje Natural. Un esquema pre-conceptual es una representación intermedia entre las especificaciones textuales en lenguaje natural ("modelos verbales") y los diferentes esquemas conceptuales que permiten el modelamiento de una pieza de software [94].

Este modelo se caracteriza por:

- Proceso de analizar los aspectos de interés de una organización y la relación que tienen unos con otros.
- Resulta en el descubrimiento y documentación de los recursos de datos del negocio.

Sus objetivos son:

- Registrar los requerimientos de datos de un proceso de negocio
- Permite observar: Patrones de datos y el uso potencial de los datos

Debido a que el modelo pre-conceptual es clara para los analistas y es sumamente útil para el proceso de elicitación de requerimientos, se han desarrollados trabajos en los que se plantean un conjunto de actividades, que se encarga de la conversión automática de texto en Lenguaje Natural en modelos

pre-conceptuales, que han sido trabajados en inglés, alemán e italiano con algunos elementos como:

- Un intérprete sintáctico que permite asignar una categoría lingüística a cada frase y trazar los árboles sintácticos respectivos.

- Un analizador semántico para categorizar y determinar los diferentes roles.

- Léxicos propios del dominio de la aplicación, con el fin de limitar el lenguaje disponible para él.

- Esquemas pre-conceptuales de diferente índole, que permiten realizar un paso intermedio entre la especificación en LN y el esquema conceptual definido.

- Un conjunto de reglas heurísticas, que permite determinar cuál de los elementos pertenecientes a un esquema conceptual se puede traducir un elemento dentro del esquema pre-conceptual definido o, en su defecto, de la especificación en LN.

Algunas herramientas y métodos son:

5.4.5.1 RADD. El proyecto, denominado **RADD** (Rapid Application and Database Development) es una metodología para la extracción del diagrama entidad-relación [95] a partir de las especificaciones en lenguaje natural:

(1) Inicia con un texto en LN, en alemán que se ingresa a un analizador sintáctico.

(2) RADD usa un modelo lingüístico que inserta un nivel semántico entre los niveles sintáctico y conceptual, identificando el significado de una frase mediante los roles semánticos asociados con los verbos.

(3) Finalmente utiliza una serie de reglas heurísticas para realizar la conversión del resultado del análisis sintáctico-semántico en los diferentes elementos del modelo entidad-relación.

Esta herramienta (**RADD**) se basa en un programa en PROLOG, que activa una serie de preguntas dependiendo del nivel de análisis que se haya alcanzado con el texto ingresado. La activación de las preguntas al diseñador se produce con un análisis sintáctico incompleto o un modelo de diseño incompleto, realizando preguntas de contenido ("¿Existen más detalles sobre la aplicación?"), clarificación lingüística ("¿Cómo se realiza el actor-prestar-?") y clarificación pragmática ("¿Cómo se caracterizan los –libros-?"). Con base en las respuestas suministradas también en lenguaje natural por parte del diseñador, se completa el modelo.

Los resultados de presentan en un modelo textual con la siguiente nomenclatura [95]:

Entity (EName): describe una entidad con el nombre Ename.

Relchip (RName, EName1, [EName2]) describe una relación RName entra la entidad EName y a lista de entidades (EName2 describe un conjunto de entidades).

Attre (EName, AName): la entidad EName tiene un atributo AName.

Attrr (RName, AName): la relación RName tiene un atributo AName.

Keycand (EName/RName, AName): el atributo AName es una clave candidata de la entidad EName o la relación RName.

Cardcand (NR, Rname, EName, MinCard, MaxCard): la relación RName tiene unas cardinalidades MinCard: MaxCard correspondientes a la entidad EName en el extremo NR.

Inclcand (EName1, EName2): describe una dependencia de inclusiones de dos entidades (EName1 y EName2, donde un EName1 incluye un EName2).

Exclcand ([EName]): describe una lista de entidades excluidas EName.

5.4.5.2 Proyecto CIRCE. Usa una herramienta llamada **CICO** [96], la cual usa un enfoque de lógica difusa para realizar el análisis sintáctico de las frases; además dentro de este proyecto se puede hacer un análisis de la completitud, consistencia y corrección de los textos en LN, realizando las siguientes funciones:

- Elicitación de requisitos (determinación de los principales conceptos del dominio)
- Selección de requisitos (para reducir la complejidad)
- Identificación de conflictos entre requisitos y forzar un buen estilo en los requisitos [97].

En [98] se muestra la forma como el CICO parser puede llegar a generar también diferentes tipos de diagramas a partir de especificaciones en lenguaje Natural

5.4.5.3 Proyecto CYRE. Cyre [99]presenta un trabajo un trabajo aplicable en el dominio particular del desarrollo de Sistemas Digitales, en el cual combina los diferentes diagramas que se puedan trazar para este desarrollo en particular (diagramas de bloques, diagramas de flujo de datos, estructura de datos, diagramas de flujo, diagramas de estado y diagramas de tiempo), con especificaciones expresadas en lenguaje natural.

Usa la herramienta **ASPIN** (Automatic Specification Interpreter), toma como entradas los modelos definidos y la especificación en lenguaje Natural y los convierte en grafos conceptuales (que son especies de redes semánticas), que emplean un lenguaje de representación común, con el fin de integrarlos y representarlos de manera gráfica para que el usuario pueda verificar inconsistencias y omisiones.

Un ejemplo particular de la aplicación del lenguaje definido para los grafos conceptuales en la oración “Un programa es ejecutado por el procesador“, tiene la forma siguiente [99]:

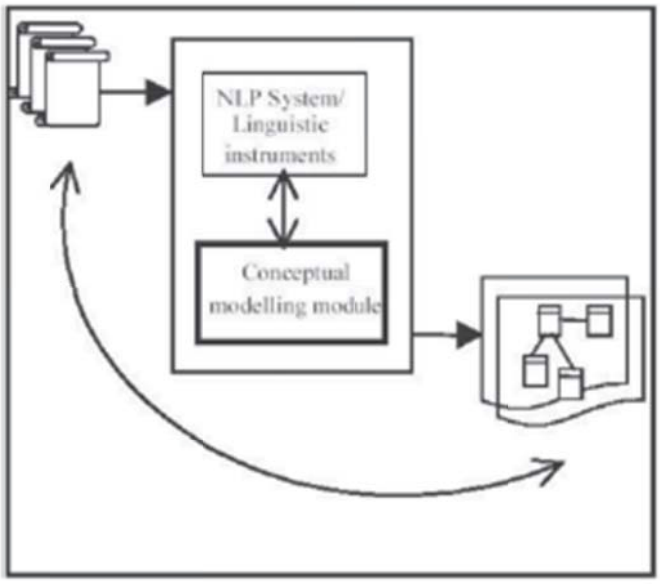
[execute]-
(agnt : by) –> [processor : #]
(opnd) – > [program : *]

Para lograr este resultado, ASPIN realiza un análisis sintáctico, en el que básicamente se determinan los verbos y los sustantivos de las especificaciones, tomando como restricción aquellos pertenecientes al dominio de desarrollo de sistemas digitales, y luego de un análisis semántico que utiliza la teoría de roles semánticos [100]. Para los demás modelos,

5.4.5.4 NL-OOPS (Natural Language Object – Oriented Product System). Es un sistema propuesto por [101] y que se basa en un sistema de procesamiento del LN denominado LOLITA (Large-scale Object-based Language Interactor Translator and Analyser), el cual contiene una serie de funciones para el análisis del Lenguaje natural, con el fin de detectar incluso ambigüedades en el texto que sirve de entrada para el análisis.

NL-OOPS entrega como resultado un conjunto de las clases candidatas y las posibles instancias, atributos y métodos de las mismas. Utiliza para la entrega de resultados una estructura en forma de árbol, porque no se establecen en él las diferentes relaciones que pueden estar presentes en el diagrama (agregaciones, generalizaciones, dependencias, etc.). El resultado de este análisis sirve como punto de partida para la complementación del diagrama de clases [102].

Figura10. Propósito de NL-OOPS. Tomado de [94]



5.4.5.5 KCPM (Klagenfurt Conceptual Predesing Model)

Fue elaborado en la Universidad de Klagenfurt, en Austria, como parte del proyecto NIBA, que busca la compresión de textos en lenguaje natural en alemán restringido para convertirlo en esquemas conceptuales que apoyen la producción automática de software.

El proyecto cuenta con las siguientes herramientas:

- Un analizador sintáctico basado en NT(M)S (Morfosintaxis teórica del lenguaje Natural), que posibilita la realización de árboles sintácticos.

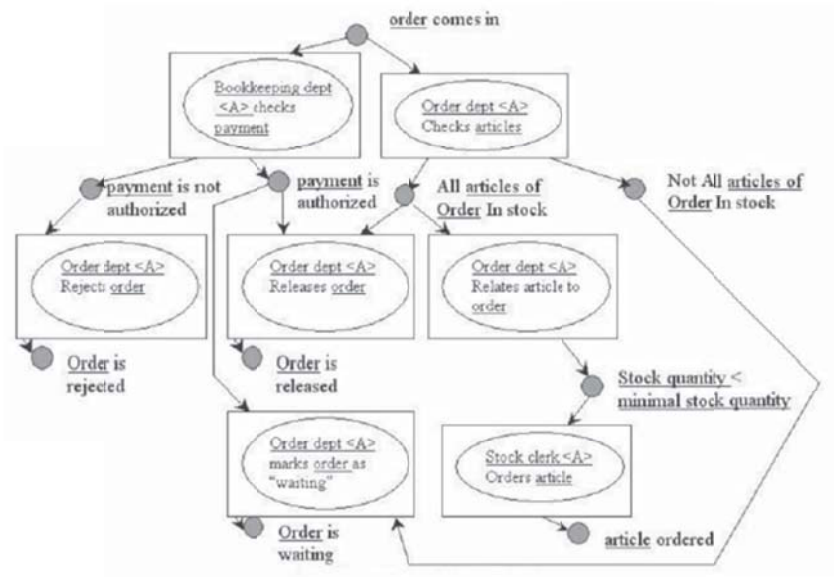
- Un intérprete semántico que determina las distintas palabras que acompañan a un verbo.

- Una herramienta de KCPM, la cual puede realizar la traducción de oraciones, analizada de manera sintáctica y semántica, en los denominados esquemas pre-conceptuales, definidos como esquemas gráficos o tabulares que posibilitan la construcción de los diferentes esquemas conceptuales, disponibles para un determinado problema de desarrollo de software.

La herramienta KCPM es diferente según el tipo de esquema conceptual a construir. Cuando se trata de esquemas de tipo estructural (diagrama de clases UML), se emplean tablas de conversión que categorizan las palabras de una frase de “tipo cosa” (elementos que posteriormente pueden ser clases o atributos) y “tipo conexión” (los cuales posiblemente pueden ser clases o incluso atributos). Cuando se trata de esquemas de comportamiento, se emplean diagramas pre-conceptuales, que categorizan los elementos en actividades o acciones, propiedades o estados, eventos, finales de actividad y restricciones (Figura 11).

En ambos casos se utiliza un conjunto de reglas heurísticas que permiten realizar la conversión de los diferentes modelos pre-conceptuales (tabulares o gráficos), en los respectivos modelos conceptuales estructurales o comportamentales.

Figura 11. Esquema pre-conceptual de la Univ. De Klagenfurt para el trazado de diagrama de actividades. Tomado de [94]



En el idioma español se han comenzado a realizar estudios sobre modelos conceptuales, que poseen un alto contenido textual. No se conocen trabajos que tomen como punto de partida el lenguaje natural en español para la construcción de esquemas conceptuales.

En cuanto a proyectos realizados para la interpretación de requisitos no funcionales de la Organización en Lenguaje Natural se tiene:

5.4.5.6 PROYECTO KAOS (Knowledge Acquisition in autOmated Specification). Incorpora requisitos no funcionales, expresados en lenguaje Natural, en un esquema conceptual construido a partir de un lenguaje formal [103]. En este esquema, el analista debe realizar la traducción del lenguaje natural al lenguaje formal [104].

Este esquema se centra en la especificación de los objetivos de la solución informática a construir por lo que tampoco garantiza la alineación de éstos con los objetivos organizacionales.

5.4.6 Otras técnicas relacionadas con el análisis de documentos:

5.4.6.1 Documentos como Vectores. Aplicando lo anterior a la RI, se puede decir entonces que un documento es un conjunto de términos (palabras o frases) que representan su contenido. Estos términos pueden obtenerse mediante indización automática, o por asignación, mediante indización intelectual con utilización de un lenguaje documental externo, circunstancia que recoge el modelo de RI ampliado de la Figura 12.

Figura 12. Modelo ampliado de RI, con inclusión de un lenguaje documental. Tomada de [105]

Así, y siguiendo el modelo de Salton y McGill [106], si tenemos un conjunto *T* de 5 términos, tal que, por ejemplo,

T = (Informática, Documentación, Telecomunicaciones, Lingüística, Estadística)

para representar los temas de los documentos de una base de datos, podemos generalizar y transformar la misma expresión en:
T = (t1, t2, t3, t4, t5) (1) donde *t1*, *t2*, etc., simbolizan cualesquiera términos.

A partir de aquí podemos representar un documento como un vector que adopta, por ejemplo, la forma siguiente:

D_k = <1 1 1 0 0>

que significa que el documento *k* contiene los términos *t1*, *t2* y *t3* del conjunto *T,* lo cual se indica con respectivos 1 en las posiciones correspondientes, pero no contiene el término *t4* ni el *t5*, cosa que se indica mediante diversos 0 también en sus posiciones correspondientes.

Un vector es una estructura consistente en un número fijo de elementos (en nuestro ejemplo, 5) en la cual la posición de cada uno es significativa.

Las preguntas también pueden representarse como un vector de la misma forma. Por ejemplo:

P_h= <0, 1, 0, 1, 1>
significa que una determinada necesidad de información ha sido indizada con los términos *t2*, *t4* y *t5*, formando así la pregunta P_h.

Ahora podríamos comprobar qué documentos de la base de datos se parecen más a la pregunta P_h y ordenarlos en función de esa semejanza, estableciendo en algún punto un umbral por debajo del cual se consideraría que un documento ya no es relevante.

Supongamos, para simplificar, que tenemos únicamente dos documentos (D_i y D_j) en la base de datos, los cuales presentan los siguientes vectores respectivos:

D_i = <1, 1, 1, 0, 0>
D_j = <1, 1, 0, 0, 1>

5.4.6.2 Función de comparación. La selección del documento más parecido a la pregunta se realiza calculando cuál de los dos documentos posee más elementos en común con ella.

Existen diversas maneras de efectuar ese cálculo [106], [107]. Una de las más sencillas consiste en el sumatorio de los productos. Es decir, los dos números de cada columna se multiplican entre sí, y los resultados se suman, tal como se ilustra a continuación:

P_h=0, 1, 0, 1, 1
D_i= 1, 1, 1, 0, 0
---
0 + 1 + 0 + 0 + 0 = 1
P_h=0, 1, 0, 1, 1
D_j= 1, 1, 0, 0, 1
---
0 + 1 + 0 + 0 + 1 = 2

De acuerdo con lo anterior, en este caso el documento D_j es el más parecido a la pregunta P_h.

Si en lugar de dos documentos, en la base de datos hubiera miles, el procedimiento sería el mismo.

Como semejante cálculo podría ser muy largo, en la práctica se utilizan dos procedimientos:

- O bien se realiza primero una recuperación convencional, utilizando un OR booleano entre los términos de búsqueda, y después se aplica el algoritmo de cálculo de relevancia al subconjunto resultante,

- O bien se utiliza una técnica de ordenación de los documentos basada en espacios vectoriales y en agrupamiento por clusters (grupos estadísticamente semejantes) y centroides (elemento más representativo de cada cluster), que facilita el proceso de comparación.

En cualquier caso, el resultado de la búsqueda proporcionaría una lista de documentos ordenados según su grado de semejanza con la pregunta, ello equivaldría a ordenarlos en función de su grado de relevancia.

5.4.6.3 Ponderación e indización automáticas. La ponderación en el punto anterior otorga mayor peso a aquellos documentos que presentan el mayor número absoluto de ocurrencias de los términos de búsqueda, lo cual es mucho mejor que una ordenación al azar, pero a veces no produce buenos resultados.

Por tanto, en lugar de que cada elemento del vector asuma un valor igual a 0 (cuando el término no se asigna al documento) o igual a 1 (cuando el término sí que se asigna), podemos pensar en establecer grados, es decir, en otorgar pesos, para expresar en qué medida un documento parece referirse a un tema. Esta operación de denomina ponderación *(weighting)*.

Así, en sistemas de RI que utilizan pesos, cada elemento del vector podrá asumir un valor cualquiera entre 0 y 1; y el vector de un documento D_i puede tener esta forma:

D_i = <0.8, 0.75, 0.0, 0.9, 0.5>

Lo que significa que el término *t1* tiene un peso, en ese documento, de 0.8; el término *t2*, de 0.75; el *t3* no está asignado, etc. Cuando se utilizan pesos en lugar de los valores binarios 0 y 1, las ecuaciones anteriores permanecen igual, ya que lo único que cambia es que ahora es necesario un procedimiento adicional para calcular el peso proporcional de cada término en el documento.

5.5 CONCLUSIONES

- Existen técnicas y herramientas que apoyan la tarea de elicitación de requerimientos y que se basan en análisis de textos en lenguaje natural, análisis de textos en código fuente, análisis de las GUI's, análisis de las interacciones, análisis de los objetos de la interfaz.

- Se encontraron técnicas para el análisis de textos escritos en Lenguaje Natural, que nos permiten analizar y extraer esquemas pre-conceptuales, los cuales son de gran utilidad para los analistas en la elicitación de requerimientos. Sin embargo, no hay bibliografía para el análisis de documentos tales como memorandos, facturas, manuales, etc.

- Se realizan modelos de alto contenido gráfico dejando de lado modelos con alto contenido textual.

- Existen pocos trabajos para el español.

- No se trabajan modelos diferentes al diagrama Entidad-Relación o a los diagramas de UML.

6 SOFTWARE HEREDADO Y USO DE PROTOTIPOS GUI

6.1 INGENIERÍA INVERSA

Este término se empleó primero en el área del hardware, siendo su primer objetivo la obtención de una especificación de un sistema hardware, sin tener los diseños originales, para realizar una copia del sistema. En el área de software su objetivo es aumentar la comprensión de una aplicación existente y recuperar modelos del sistema, con propósitos de mantenimiento, evolución, mejora o migración.

El propósito de la ingeniería inversa no es el cambio o duplicación del software sino más bien su análisis y comprensión. Se puede dar en cualquiera de las etapas de un desarrollo software, obteniéndose desde los detalles del diseño hasta los requisitos que están implementados en la actualidad. En otras palabras la ingeniería inversa es el proceso de ver un software heredado en un nivel alto de abstracción y engloba varios métodos y herramientas relacionadas. Como tarea de elicitación de requisitos es recomendada por la IEEE [33].

Según el articulo New Frontiers of Reverse Engineering [34], la ingeniería inversa se ha visto como un proceso de dos pasos: extracción y abstracción de la información, durante la extracción de la información se analiza los artefactos software para obtener datos sin refinar, y luego en la etapa de abstracción se crean documentos y vistas en alto nivel de abstracción para que los usuarios puedan comprenderlos mejor.

Los recursos del proceso son los artefactos del sistema, entre estos el código, que es el más comúnmente usado, y los modelos, especificaciones o documentación. En estos últimos se busca actualizar su contenido para que refleje los cambios que se presentaron en las operaciones de mantenimiento.

Desde sus comienzos la ingeniería inversa se ha enfocado, en el análisis del código de la aplicación existente, pero esto genero algunos obstáculos en el avance del proceso, como el alto costo que tiene la comprensión del código en grandes sistemas [35], por ejemplo en algunos esfuerzos de re-documentación de sistemas heredados, el análisis del código no fue suficiente [36] ya que, como se estableció en este reporte, toma mucho tiempo analizar todo el código fuente de un gran sistema; pero lo interesante de estos proyectos es que aunque se parte del código, es para realizar la identificación de los elementos de la interfaz [37], en estos proyectos se ejecutaron análisis de la interfaz para obtener su arquitectura y comportamiento, ya que podían corresponden en cierta medida a la arquitectura y comportamiento del sistema. Esto nos muestra que en las tareas de la ingeniería inversa también se han comenzado a utilizar otros elementos del sistema como las interfaces.

También en otros estudios de la ingeniería inversa se han encontrado buenas técnicas para obtener documentos de diseño o requisitos, entre estas las que sugieren el uso de elementos gráficos para una mejor comprensión de la arquitectura software y proveer un análisis del código [38].

Según estos proyectos hay dos tipos de análisis de la interfaces, un análisis estático, centrado en el código relevante a la interfaz de usuario, en donde a partir del código fuente se extrae un gráfico de las ventanas, que actúa como soporte a la recuperación de la arquitectura [39]. Y un análisis dinámico que consiste en la ejecución del programa, obteniéndose igualmente información de la arquitectura y el comportamiento del sistema pero con la ventaja de no tener que lidiar con extensos volúmenes de código [40, 41, 42].

También la información dinámica (de ejecución) es útil para la comprensión del comportamiento de sistemas software, pues veremos requisitos justos observando el comportamiento del usuario. [RETR panel]

6.2 PROTOTIPOS GUI

Los prototipos son pequeñas implementaciones de un sistema software que ayudan en su diseño pero que además se pueden utilizar como una técnica de elicitación y validación de requisitos [43] [44] [45]. Los prototipos se definen como mecanismos de comunicación entre los clientes, usuarios y analistas, que demuestran las decisiones tomadas, para validarlas, resolviendo los problemas de comprensión que se presentan en la etapa de elicitación.

El uso de prototipos es una de las técnicas más eficientes de elicitación. Este método es especialmente útil cuando existen dudas acerca de los objetivos de la futura aplicación, y cuando las opiniones del usuario son necesarias en una fase temprana del proceso de especificación [46, 47].

Algunas de las técnicas conocidas de prototipado son los sketches [48] y el storyboard [49, 50], y son técnicas, principalmente de diseño, que muestran la secuencia de ventanas y las acciones asociadas a los cambios de ventana a ventana de una manera informal, sin codificar y más flexible.

Aunque se consideren como técnicas de diseño se ha comprobado [51] que estas técnicas visuales suelen tener más éxito que otras en la comunicación con los clientes o usuarios (stakeholders) en las fases de elicitación de requisitos, es decir, estos modelos visuales siempre originan discusiones entre los stakeholders, que derivan en la aceptación o rechazo de los requisitos ya identificados y en la generación de nuevos requisitos, esta comunicación de requisitos se lleva a cabo mediante los esbozos de la interfaz. Es así como se plantean como una excelente fuente de realimentación, recolección de requisitos y colaboración a la continua mejora del modelo de conocimiento del producto requerido [52, 53].

Dentro del resumen de las técnicas de elicitación según [54] se encuentran los Storyboards, que se definen como colecciones de pantallazos e imágenes que muestran cómo se verá el sistema final. También se encuentra una clasificación de esta técnica en dos tipos:

- Storyboards pasivos que son simples colecciones de imágenes que la ingeniería de requisitos muestra en secuencia al usuario.
- Los Storyboards activos que son más parecidos a los prototipos de sistema.

Un prototipo activo puede ser navegado, es decir, el usuario se puede mover dentro de la secuencia de imágenes en un orden lógico y definido, pero sigue siendo solo una colección de imágenes que muestran el sistema final, además, en general los Storyboards no son denominados interactivos ya que no requieren ni aceptan datos. [53]

Muy similar a la anterior, pero ya referente a un prototipo, es la clasificación según la funcionalidad [55], están:

- Los prototipos de Baja Fidelidad: que se limitan a un conjunto de dibujos (por ejemplo, una presentación de escenarios) que constituye una maqueta estática, no computarizada y no operativa de una interfaz de usuario para un sistema en desarrollo, y
- los prototipos de Alta Fidelidad son un conjunto de pantallas que proporcionan un modelo dinámico, computarizado y operativo de un sistema en desarrollo.

Los prototipos de baja fidelidad pueden ser usados como una herramienta de escucha (comunicación) activa ya que a menudo cuando las personas no pueden articular una necesidad particular en abstracto, si pueden evaluar rápidamente si

un enfoque de diseño puede manejar su necesidad. Los prototipos son hechos más eficientemente con bocetos (sketches) de interfaces y storyboards.
En cuanto a su construcción hay dos tipos [11, 53]:

- los prototipos desechables sirven para una única presentación, que valida o no lo expuesto por el prototipo (ya sea requisitos, diseños, etc.), son construidos rápidamente usando atajos y tecnologías alternas para ganar conocimiento y demostrar ideas y no se tienen en cuenta para otro prototipo posterior, y
- Los prototipos evolutivos se construyen usando la arquitectura del sistema final con el propósito de que este sea eventualmente desarrollado, en forma iterativa, en el sistema real.

En cuanto al alcance también podemos hablar de clases de prototipado [53], están:

- Los prototipos horizontales que incluyen un amplio rango de funcionalidades del sistema.
- Los prototipos verticales que exhiben solo unos pocos requisitos del sistema.

Dentro de los diferentes métodos que usan prototipos para la fase de análisis de requisitos en un entorno particular como el web, está la propuesta de HFPM [57] en cuya primera fase denominada de modelado de requisitos esta la tarea de modelar la interfaz de usuario y para ello, propone el uso de sketches y prototipos que permiten presentar los datos al usuario, a manera de validación de los requisitos recopilados. También está el método UWE (UML-Based Web Engineering) que propone como técnicas apropiadas para la captura de los requisitos de un sistema web, entre otras técnicas, los casos de uso y los escenarios y para la validación propone los prototipos.

El uso de los prototipos para representar la navegación del sistema, la visualización de los datos y la interacción con el usuario se da también en la propuesta NDT (Navigational Development Techniques) en donde se definen unos requisitos de interacción, que es representan mediante prototipos de visualización.

En Design-driven Requirements Elicitation, el proceso se basa en el uso de prototipos para ayudar al cliente en la exploración de las posibles soluciones y de los problemas que tienen que ser resueltos. Los usuarios o clientes definen sus requerimientos basándose en la observación o trabajo con estos prototipos. [58] [59]

Por último para el modelo DUTCH (Designing for Users and Tasks from Concepts to Handles) la utilización de alguna clase de escenario o prototipo será a menudo la mejor manera de confrontar al usuario final con la solución propuesta. Acá se expone que un escenario permite representar el nuevo modelo de tareas previsto para situaciones futuras de uso y un prototipo permite experimentar con los elementos seleccionados de los diferentes modelos que se desarrollen [51].

6.3 ESCENARIOS

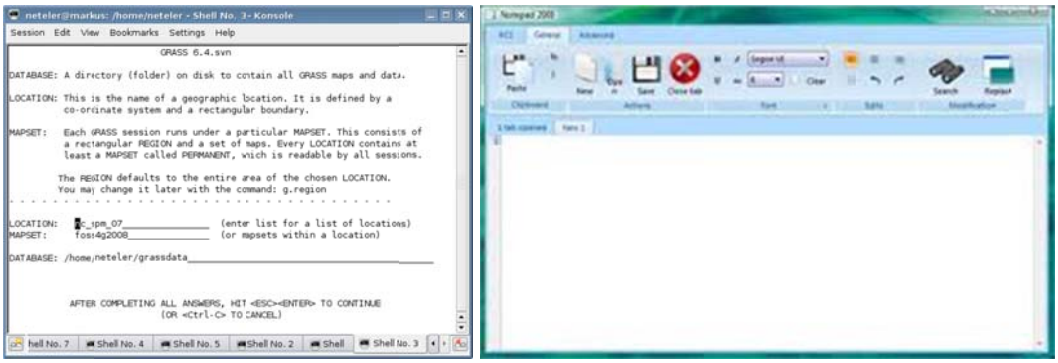
Dadas las potenciales dificultades con los usuarios y clientes, relacionadas con la comunicación de sus necesidades, la elicitación de las especificaciones de requisitos a partir de escenarios operacionales se muestra como una buena opción que requirió de ciertas consideraciones, primero, se definió el escenario como una secuencia en el tiempo de eventos de interacción entre diferentes agentes en el contexto restringido de conseguir algún propósito específico [61]. Esta definición restringe a que el escenario solo capturara una instancia particular del comportamiento del sistema.

También se puede hacer la distinción entre escenarios positivos y negativos, entendiéndose los primeros como aquellos del comportamiento deseable en términos del usuario, y los segundos como el no deseable.

6.4 ESTADO DEL ARTE

6.4.1 Técnicas basadas en objetos de interacción. Consiste en el análisis de los elementos de la interfaz para determinar los objetos, clases, tablas que debe soportar el nuevo sistema. Los elementos encontrados varían de acuerdo a la naturaleza de la interfaz, por ejemplo, algunas interfaces carecen de elementos propios como la interfaz basada en comandos, en cambio las interfaces graficas se constituyen de elementos tales como formularios (entradas de texto, selección, marcado), botones y menús.

Figura 13. Interfaces textual y grafica – Tomado de [64] [66]



Dada la variedad de técnicas de análisis, se tienen unas denominaciones de los elementos de las interfaces gráficas, que pueden servir a manera de correspondencia. Entre los elementos de la interfaz llamados CIO (objetos de interacción concreta) y otra representación como los AIO (objetos de interacción

abstracta) se da una relación que esta descrita en el artículo [69], también la relación entre los elementos HTML y las declaraciones textuales en lenguajes como XIML (eXtensible Interface Markup Language) se puede encontrar en [67] y los llamados widgets también tienen correspondencias en un lenguaje denominado LEMD (Designer's Message Specification Language) [68], todas estas correspondencias para asistir el análisis.

La correspondencia entre CIO y AIO define una clasificación que muestra los diferentes tipos de interacciones, es decir según [69], existen 6 tipos de interacción o conjuntos de objetos de interacción abstracta, que son:

Tabla 3. Tabla de objetos de interacción abstractos

Conjunto AIO	Elementos (algunos) AIO
Objetos de acción	Menu, menu item, menu bar, drop-down menu, cascade menu, submenu,...
Objetos de scrolling	Scroll arrow, scroll cursor, scroll bar, frame
Objetos estáticos	Label, separator, group box, prompt, icon
Objetos de control	Edit box, scale, dial, check box, switch, radio box, spin button, push button, list box, dropdown list box, combination box, table...
Objetos de dialogo	Window, help window, dialog box, expandable dialog box, radio dialog box, panel...
Objetos de respuesta	Message, progression indicator, contextual cursor

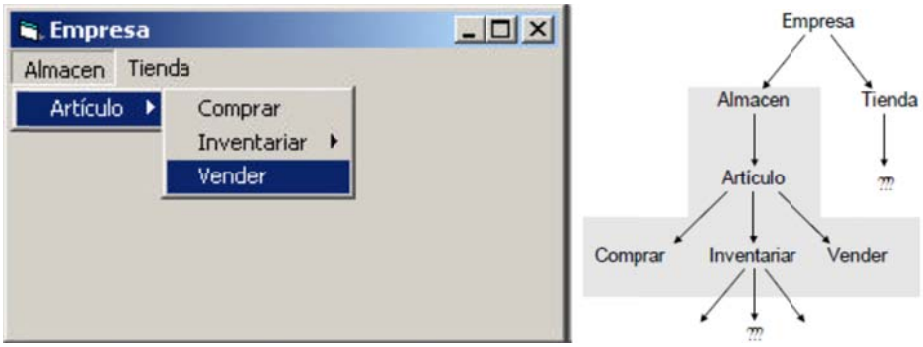
Los conjuntos de AIO (objetos de interacción abstracta) que pueden ser especificados sobre modelos conceptuales orientados a objetos [70] como en las herramientas JANUS, TRIDENT y otras similares como Genova, MOBI-D, Wisdom y el mapeo de los signos de las Interfaces de usuario o widgets a declaraciones en LEMD [68] que luego pueden ser aproximadas a la notación de casos de uso o

modelos de tareas, son investigaciones resultado de un análisis estático de los elementos de la interfaz.

Otra técnica para otro tipo de interfaces, las interfaces Web, y que busca la especificación independiente de la plataforma mediante la declaración de los elementos HTML en un nuevo lenguaje, el XIML, se define en [67] junto con herramientas que sirven a ese objetivo como PEGASUS y DESK. XIML representa a la interfaz en función de cinco dimensiones las cuales son: tareas, dominio, usuario, presentación y dialogo.

Los botones y menús tienen mayor significado en ejecución (justo para un análisis dinámico), pero aun así pueden ser analizados estáticamente, como en la técnica vista en "*Ingeniería de Requisitos aplicada al modelado conceptual de interfaz de usuario*" [78], que plantea una correspondencia entre los casos de uso, en representación del modelo de requisitos y los menús, en la forma de un árbol de refinamiento de funciones. Esta técnica se centra en el Árbol de Refinamiento de Funciones que combinado con el modelo de casos de uso se puede tomar como el patrón conceptual de jerarquía de acciones que es una abstracción útil para construir un árbol que exponga al usuario final la funcionalidad del sistema.

Figura 14. Menú generado para una vista de caso de uso – Tomado de [78]



También como agrupaciones, los elementos de una interfaz pueden simbolizar diferentes unidades como las descritas en [71], en la técnica se presentan las unidades de interacción y presentación, además de mapas de navegación que se corresponden con clases del modelo conceptual.

Esta técnica orientada al diseño de interfaces de usuario [71], busca la especificación de patrones que reflejen las primitivas dentro del modelo conceptual de una aplicación orientada a objetos. Es decir teniendo un esquema de clases (atributos, métodos y relaciones entre las clases), se pasa a identificar las primitivas relacionadas a cada una de las clases, para luego construir con ellas los escenarios del sistema (utilizando las agrupaciones de primitivas en UI-unidades de interacción o UP-unidades de presentación)

Según [71] se identificaron cinco primitivas recurrentes en el desarrollo de interfaces de usuario orientadas a objetos.

- Criterio de filtrado o selección (como buscar)
- Criterio para ordenar (como ordenar)
- Display Set (que propiedades se deben mostrar)
- Navegación (que información adicional se ha de consultar)
- Acciones (que acciones se pueden llevar a cabo sobre los objetos)

Y de las agrupaciones o unidades de interacción se presentan cuatro subtipos o patrones [71]:

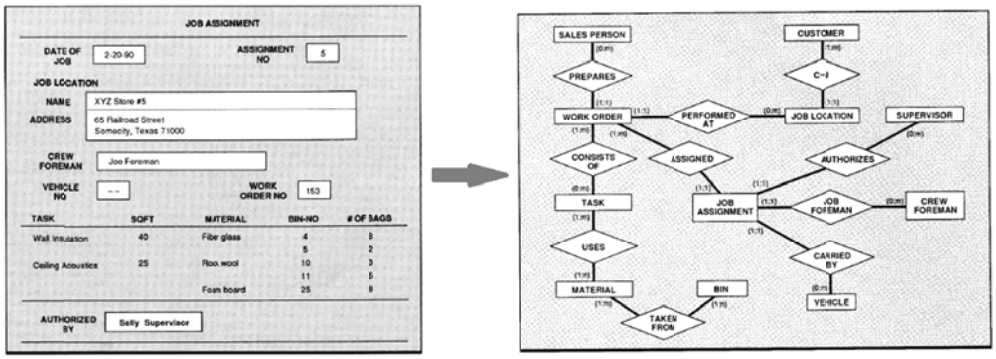
- Instance Pattern (muestra un objeto)
- Population Pattern (muestra un grupo de objetos que pertenecen a la misma clase)
- Master/Detail Pattern (muestra información relacionada con una única unidad)

- Service Pattern (le permite a los usuarios lanzar los servicios)

Usando un caso de uso e integrándolo con el patrón de interfaz de usuario obtenido, un analista puede especificar un mapa de navegación. Esta técnica presenta la correspondencia entre la navegación y distribución de los elementos de la interfaz gráfica con el modelo conceptual lo que ayudaría en la elicitación y análisis de requisitos, aclarando las decisiones de diseño de la interfaz gráfica en términos de los modelos de especificación.

Por otra parte, los formularios son más susceptibles de un análisis estático, esto es lo que aprovecha la técnica encontrada en [72], que pasa de los formularios de la interfaz de usuario a Diagramas E-R. Esta técnica hace inferencias acerca de la estructura jerárquica de un formulario así como de las dependencias funcionales entre los campos del formulario permitiéndole extraer diagramas de E-R.

Figura 15. Ejemplo de forma y su diagrama entidad relación.



Aparte de la estructura de los formularios también se puede utilizar la interacción con el usuario para determinar escenarios y modelos UML como en [73] donde se especifica la metodología FORE. Esta metodología consiste de 5 fases: análisis

del uso del formulario, división de objetos del formulario, modelado de la estructura de objetos, diseño de escenarios e integración de modelos.

En la primera fase se recoge toda la información estructural y de interacción del formulario, para que en las siguientes fases se realice el modelado conceptual orientado a objetos. Las cuatro fases restantes utilizan el conocimiento recopilado en la primera para obtener un diagrama de clases inicial de la aplicación, por ejemplo en la segunda fase, división de objetos del formulario, se tiene como objetivo seccionar la información recopilada en unidades semánticas de acuerdo al tipo de entradas, luego en la fase de modelado de la estructura de objetos, se identifican los objetos y se definen en términos de nombre, atributos y relaciones estructurales básicas. El objetivo de la fase de diseño de escenarios es un diagrama del escenario de acción de procesos de objetos a partir de la información de las operaciones realizadas y obtenidas en la primera fase, por último, la fase de integración, integra y mejora los modelos a un nivel más alto.

En esta última técnica se puede ver que se pasa de un análisis estático de los componentes de una interfaz, a incluir información acerca del uso e interacción que se observa, para obtener más información en la etapa de análisis.

6.4.2 Técnica de análisis de interacciones. Estas técnicas se basan en la observación y grabación de las interacciones de la interfaz con el usuario como ejemplos del comportamiento y uso del sistema. Se pueden definir como orientadas a un análisis dinámico del sistema usando el componente más visible del sistema, la interfaz.

Una primera técnica basada en la ejecución, grabación de la interacción de la interfaz se define en [40] y se aplica en las interfaces de comandos, aquellas con las que se comenzó una interacción más visual entre el computador y el humano. En este análisis se busca información del uso que se hace del sistema

identificando el comportamiento habitual y aquellos errores o fallas en los procedimientos, tanto de parte del sistema como del usuario. En la práctica es mucho más sencilla la grabación y análisis de estas interfaces que las interfaces gráficas.

Al diseñar sistemas adaptables se utilizan técnicas de observación que capturan la frecuencia de uso [74], estas técnicas consisten en obtener los menús o funciones más utilizadas (gracias a sub-tareas que monitorean la frecuencia de utilización de estos menús) y presentarlas más a la vista que aquellas que no son tan utilizadas, esta medida de la frecuencia de uso es un indicador de las funciones que son utilizadas y de las que no, esta técnica servirá para la identificación de las funciones que deben ser tenidas en cuenta en el nuevo sistema.

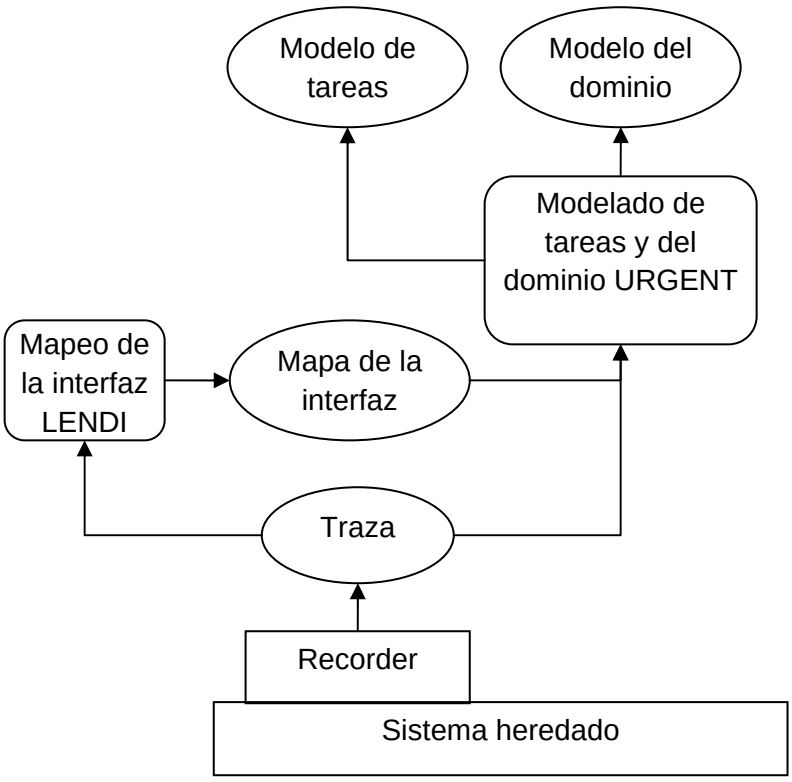
Por otra parte un proyecto distinto, el proyecto CelLEST, se enfoca en la migración de sistemas basados en transacciones, y ha desarrollado varias herramientas para esto [75]. El objetivo del proyecto CelLEST es la comprensión de la información y la lógica del proceso que incorpora el software heredado mediante la ingeniería inversa de la interfaz del sistema.

CelLEST se concentra en “los sistemas basados en transacciones desarrollados para mainframes, cuyo objetivo es prestar apoyo a tareas tales como la entrada de datos, consulta de base de datos y generación de informes, no realizan largos cálculos, pero a menudo interactúan con sus usuarios para obtener la entrada o la salida de retorno de sus transacciones. Como resultado de ello, sus interfaces exponen a sus usuarios una gran cantidad de la información almacenada en su interior y repositorios de su organización. También corresponden bastante fielmente a las diferentes tareas que actualmente se llevan a cabo.” características de un sistema que vemos a menudo en el entorno objetivo de este análisis.

El método de la ingeniería inversa de la interfaz en CelLEST consiste en dos tareas, el mapeo de la interfaz y el modelado de tareas y del dominio. En la primera tarea, se realiza un mapa de la interfaz, que presenta las pantallas y transiciones entre ellas. Una pantalla en CelLEST se define como una “unidad distintiva de presentación de información, disponible para los usuarios en un punto particular del proceso de interacción con el usuario” y una transición se realiza por medio de una secuencia de acciones de usuario, tales como pulsaciones del teclado y movimientos del cursor.

La tarea de mapeo de la interfaz la realiza la herramienta LENDI (LEgacy Navigation Domain Identifier) que identifica y agrupa las pantallas recopiladas por el Recorder obteniéndose un gráfico dirigido llamado gráfico de interfaz. La siguiente tarea de modelado de tareas y el dominio se lleva a cabo por la herramienta URGENT (User interface ReGENeration Tool), y utiliza el mapa o grafico de interfaz para construir un modelo abstracto de la interacción del usuario con el sistema. La sub-estructura del proyecto CelLEST que se encarga de la ingeniería inversa se muestra en esta figura:

Figura 16. Parte de la estructura del proyecto CelLEST



Al mismo tiempo, en el proyecto CelLEST, se define una técnica de minería de patrones [32] con el fin de identificar los escenarios de uso correspondientes a los requerimientos funcionales del sistema heredado, los cuales se deben tener en cuenta para no violar la integridad de los usos presentes. Este método se desarrolló debido a los problemas de pérdida de la documentación de los requerimientos y funcionamiento en sistemas heredados. El método analiza trazas de la interacción del usuario del sistema en tiempo de ejecución para descubrir los patrones recurrentes más frecuentes, estos patrones corresponden a la actual funcionalidad ejercida por los usuarios del sistema representada como escenarios de uso.

Figura 17. Segmento de la traza de un sistema

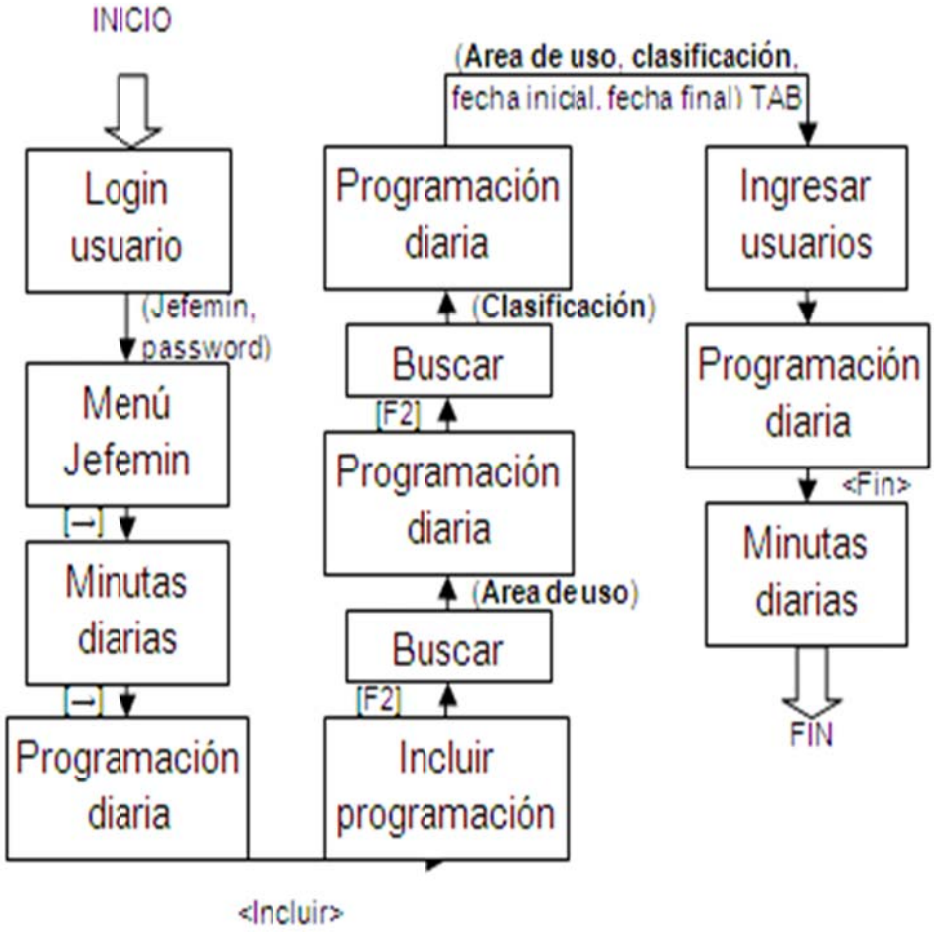
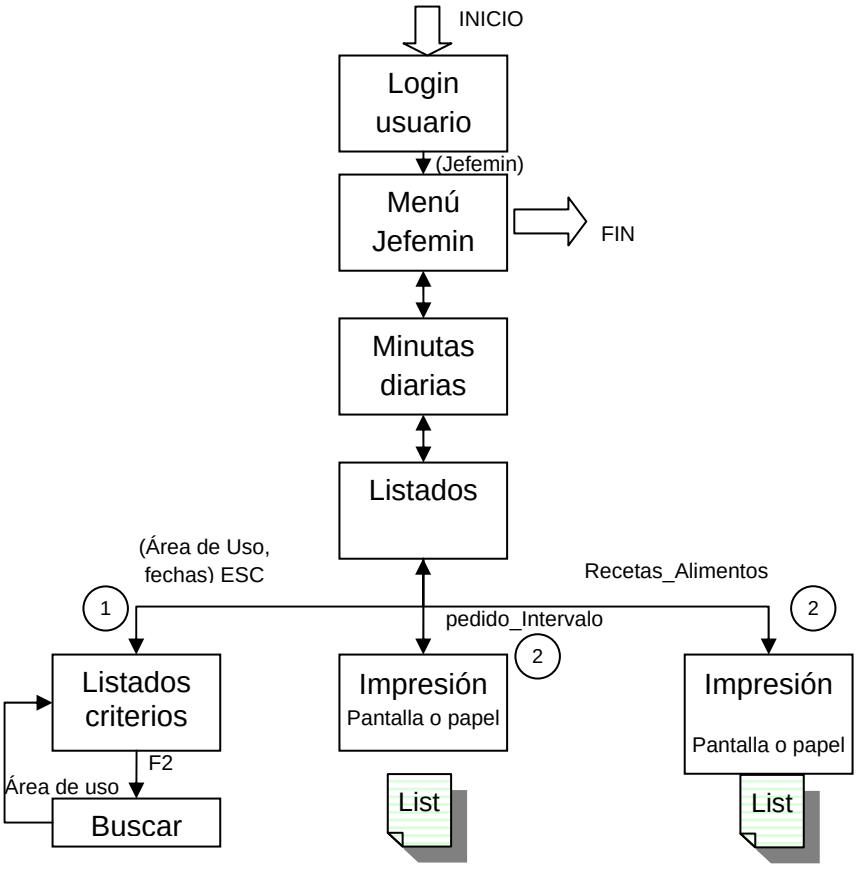


Figura 18. Segmento del diagrama de transición de estados



Los resultados del método son diagramas que representan la funcionalidad del sistema y la interacción de este con los diferentes usuarios llamados patrones de interacción. Estos diagramas se pueden agrupar según los procesos del negocio relevantes, lo que nos daría la información para complementar el análisis de casos de uso y procesos del negocio.

Por último una técnica de minería Web para el proceso de comprensión de una aplicación Web se detalla en [76] la cual consiste en una heurística que ayuda en

la identificación de las clases importantes durante la ejecución de la aplicación. La técnica propuesta se puede ver como un proceso de 4 pasos:

- Definición del escenario de ejecución, que es en donde la funcionalidad de la aplicación aparece
- Ejecución del escenario en donde se graba la traza de eventos que está compuesta por llamadas y retornos a métodos presentes en la ejecución, la salida de este paso es el diagrama compacto de llamadas, que a su vez es la entrada del siguiente paso.
- Minería de datos, examinando la traza de eventos se descubren las clases que juegan un papel importante en la ejecución.
- Interpretación de los resultados

Esta técnica combina el estudio de las trazas de interacción entre las diferentes pantallas del sistema, teniendo en cuenta un escenario seleccionado y heurísticas que ayudan a descubrir una primera perspectiva del modelo conceptual durante la ejecución de la aplicación.

6.4.3 Técnicas de obtención de casos de uso a partir de las interacciones. La siguiente técnica se concentra en la obtención de casos de uso a partir de las trazas de interacción del usuario con el sistema y también en la presentación más formal del modelo de requisitos.

En el artículo [42], se amplía lo expuesto para las técnicas de análisis de las interacciones y se recurre a las trazas de interacción del usuario para obtener los casos de uso que corresponden a las funcionalidades presentes en el software heredado. Como los patrones de interacción ya han sido obtenidos por la técnica anterior, son valorados teniendo en cuenta las tareas de interés para el usuario, ya que, un caso de uso es la secuencia de acciones realizadas por un actor (que

puede ser un usuario, otro sistema o una pieza de software). Estos patrones seleccionados son candidatos a casos de uso, ampliando lo mostrado con diagramas de actividad e información textual. Un ejemplo se muestra a continuación:

Figura 19. Traza y diagramas de transición de estados

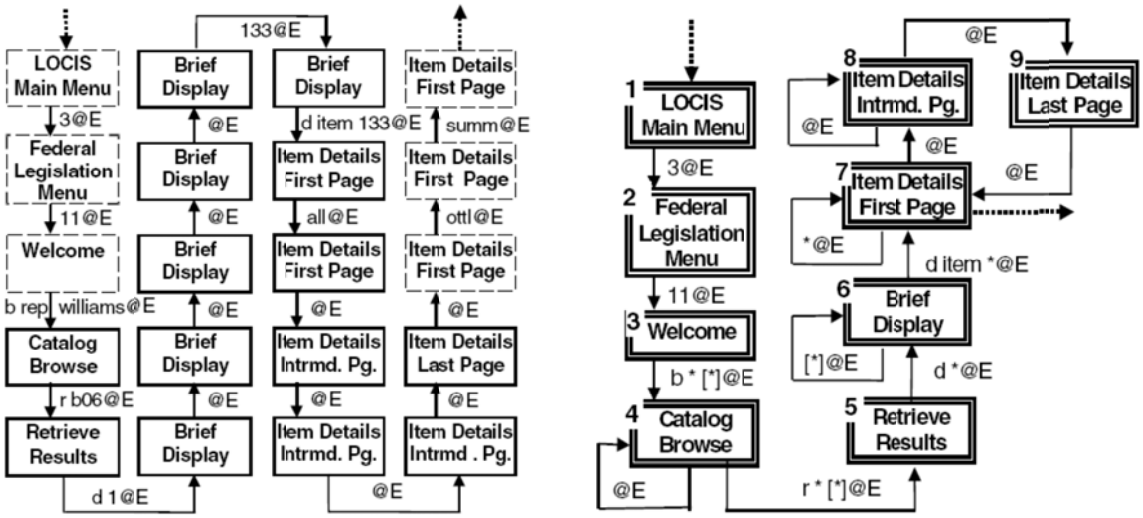


Figura 20.Patrón de caso de uso.

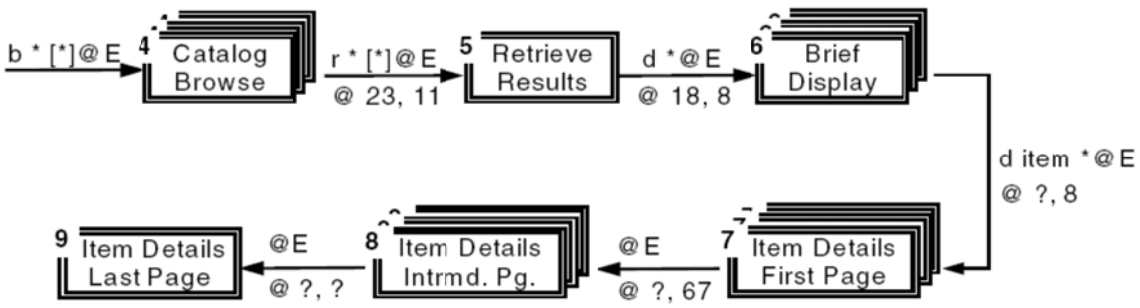
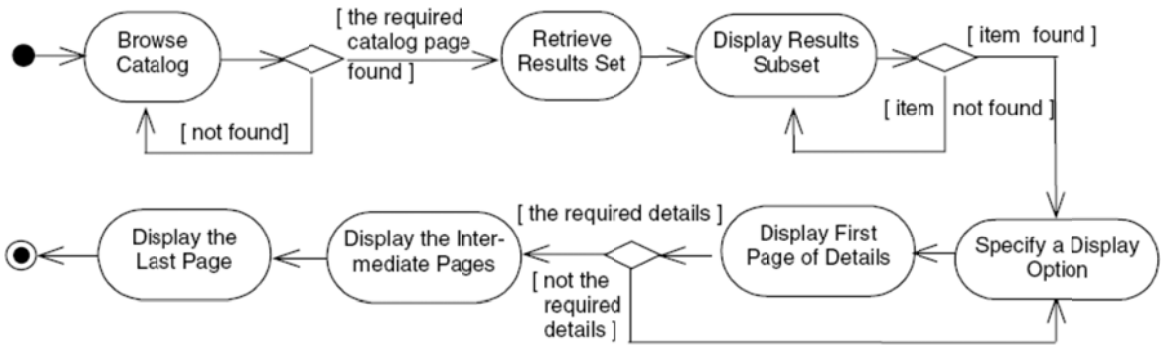


Figura 21. Modelo UML del caso de uso.



El análisis de las trazas de interacción del usuario con sistemas heredados para identificar los patrones recurrentes de actividad, correspondientes a los casos de uso, consiste en 5 pasos:

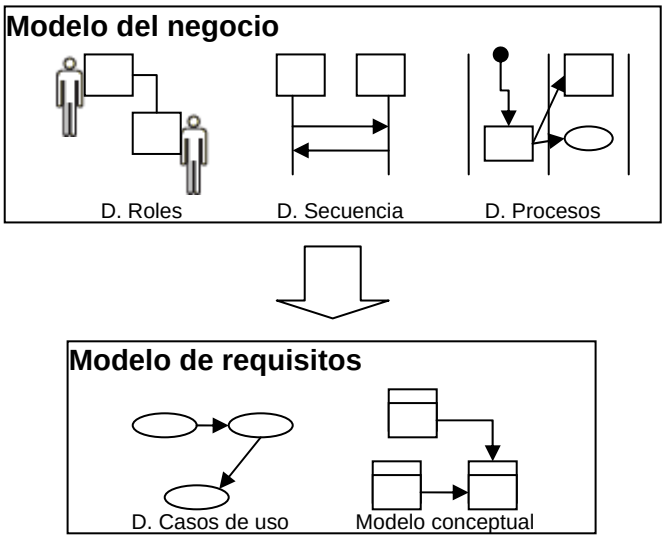
- Construcción de la sintaxis de representación apropiada para las trazas de entrada.
- Definición de un criterio para la identificación de patrones en la traza.
- Extracción de los patrones que corresponden a los criterios.
- Verificar o modificar los resultados según la realimentación con los usuarios.
- Construir el modelo de casos de uso a partir de los patrones extraídos.

Para esta técnica los casos de uso constituyen los modelos operacionales de los requerimientos funcionales de la aplicación heredada, desde la perspectiva del usuario [77].

6.4.4 Técnicas de obtención de modelos. Esta técnica se relaciona con la obtención de un primer modelo de clases del dominio del problema a partir de los modelos del negocio (diagramas de roles, de secuencia y de proceso) [26]. Se

busca elaborar el modelo de requisitos que se representara por diagramas de casos de uso y el modelo conceptual (diagramas de clases), estableciendo una correspondencia entre las entidades del negocio y las clases del modelo de análisis del sistema en cuestión [79].

Figura 22. Vista general de la técnica

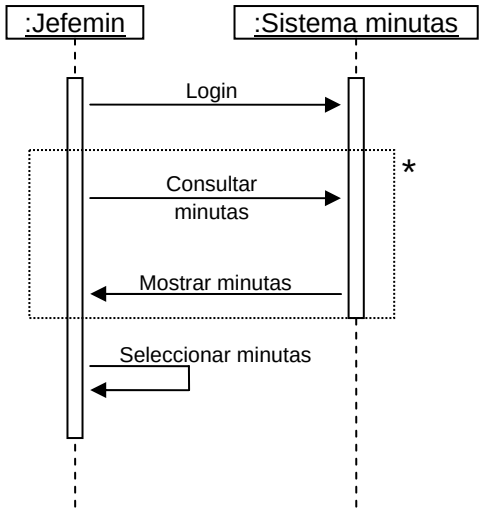


Primero se definen los procesos del negocio que son relevantes y luego los usuarios o actores que los realizan, delimitando unos casos de uso y unos roles que dan una visión global del sistema. Luego se detalla cada caso de uso estableciendo elementos internos, actividades y relaciones para cada uno.

Para realizar este detalle de los casos de uso se piensa en unos escenarios que muestren la interacción entre los roles participantes mediante diagramas de secuencia UML. En la definición de los escenarios hay que tener en cuenta no solo los cursos normales sino también aquellos alternativos. Todos estos escenarios se amplían en diagramas de proceso cada uno de los cuales constan

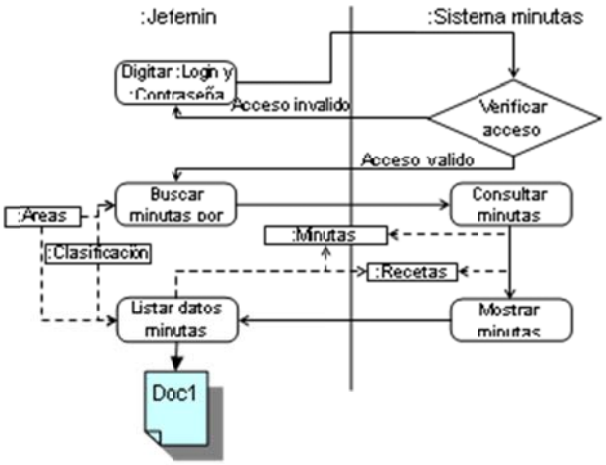
de espacios de influencia de los roles en donde están las actividades de cada uno y los datos importantes que son ‘objeto’ de la interacción.

Figura 23. Diagramas de secuencia – contiene los actores encontrados



En resumen para alcanzar un primer modelo de clases del dominio del problema a partir de los diagramas de secuencia del sistema se debe tener en cuenta los datos que fluyen entre actividades [26]. Es por eso que como no es posible ir directamente de las interfaces de usuario a este modelo conceptual, es a través de otros modelos que si son originados en el estudio de la interfaz de usuario… las clases representan los datos con los que el sistema trabajara [79].

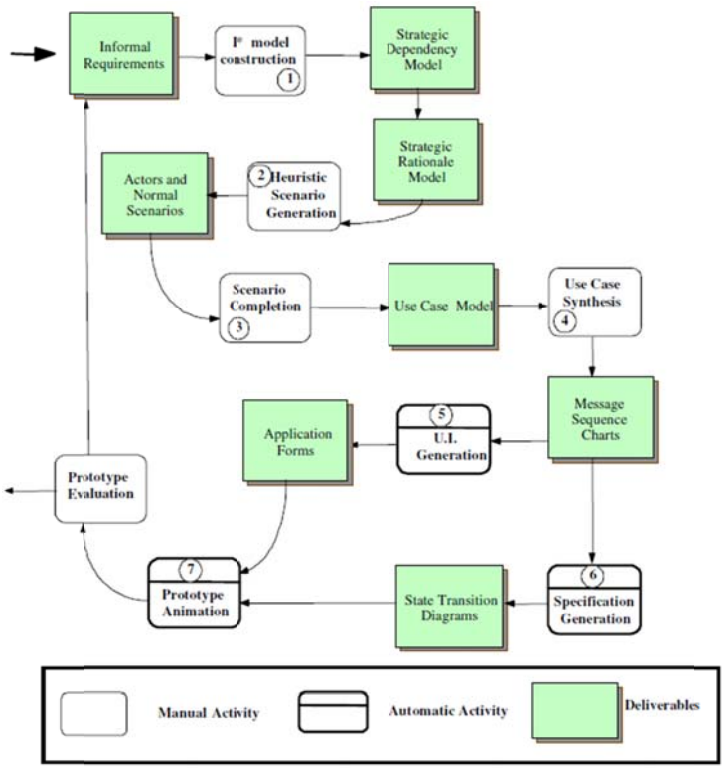
Figura 24. Ejemplo de Diagrama de proceso y Modelo conceptual



6.4.5 Técnicas de validación y realimentación del análisis. Las siguientes técnicas utilizan los prototipos o construcciones gráficas, como medio para comunicar el análisis del sistema y el diseño de la interfaz a las partes interesadas en un nuevo sistema. Se encontraron esencialmente 2 tipos: los prototipos y el storyboard.

Una primera técnica es la que se muestra en [80], aquí se utilizan los casos de uso representados como prototipos de interfaz. En esta técnica los prototipos se toman como intermediarios entre el modelo del negocio y el sistema, facilitando la validación temprana de los requisitos encontrados. En más detalle lo que hace la técnica es transformar los modelos del negocio en casos de uso representados en diagramas MSC (Message Sequence Chart) que luego son mostrados en forma de prototipos.

Figura 25. Esquema del método – tomado de [80]



El proceso general consiste de:

- Crear un modelo organizacional
- Formar un modelo inicial de los procesos del negocio
- Tras la definición preliminar de los actores organizacionales que participan en los procesos del negocio se introduce el actor sistema, para representar como este soporta las relaciones propias entre los actores organizacionales.
- Generación de los casos de uso
- Hallar las correspondencias entre el modelo del negocio y modelo de casos de uso mediante heurísticas.

- Identificar los actores del caso de uso.
- Representar las dependencias objetivo que son candidatas a casos de uso, mediante plantillas que detallan los objetivos del usuario y las responsabilidades del sistema.
- Formalización de proceso de los casos de uso y generación de componentes de la interfaz de usuario.
- Representar el caso de uso en un lenguaje no ambiguo como el MSC (Message Sequence Chart). En el diagrama se simbolizan los actores y clases, estableciendo los mensajes de intercambio de información, además se introduce el objeto de interfaz y el de control de acuerdo con la técnica MSC.
- Especificación del diagrama de secuencia de mensajes (MSC) en diagramas de transición de estados. Se comienza con la selección de cada caso de uso o su conjunto de MSC luego el analista selecciona las clases para generar el diagrama de transición de estados. Para animar el prototipo obtenido el analista selecciona por defecto el objeto de interfaz y de control de cada MSC.
- Se generan dos modelos, uno es la vista o prototipo y el otro es el modelo de navegación.

En otro artículo, se muestra a los prototipos como una técnica que integra los diferentes puntos de vista que tiene cada stakeholder, diseñador y analista sobre un nuevo sistema y además incluye el concepto de multi-fidelidad [81].

La multi-fidelidad plantea un diferente nivel de funcionalidad y visualización para cada prototipo, según las necesidades de la etapa en la que se desarrolle. En esta técnica los prototipos de IU pueden llegar a tener varios niveles de detalles, posiblemente a lo largo de su evolución, y el paso de un nivel de fidelidad a otro se realiza a través de una herramienta que se denomina UsiXML [81].

Figura 26. Diferentes niveles de prototipos – Tomado de [81]

Fig. 1. No-fi mode without labels

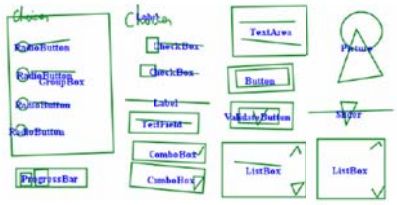


Fig. 2. Lo-fi mode for sketchingUI elements (with labels)

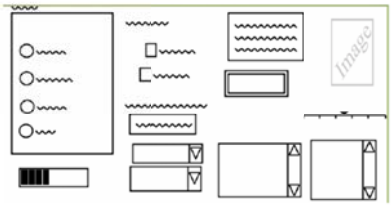


Fig. 3. Me-fi mode without labels

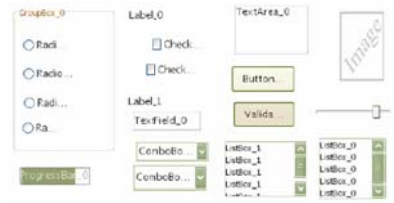


Fig. 4. Hi-fi mode without labels

Otra técnica que ayuda a identificar la funcionalidad de un sistema mientras permite que los usuarios vean y piensen en términos de la interfaz humano-computador se identifica en [49], en este artículo se señala además que un caso de uso es una herramienta muy utilizada para capturar requisitos funcionales que soporta un enfoque dirigido por escenarios los cuales permiten a los analistas concentrarse en porciones de los requisitos con buen alcance y se introduce el concepto de storyboard el cual incluye tanto las historias de usuario de los casos de uso como referencias a elementos de software y hardware, tal como la interfaz gráfica de usuario.

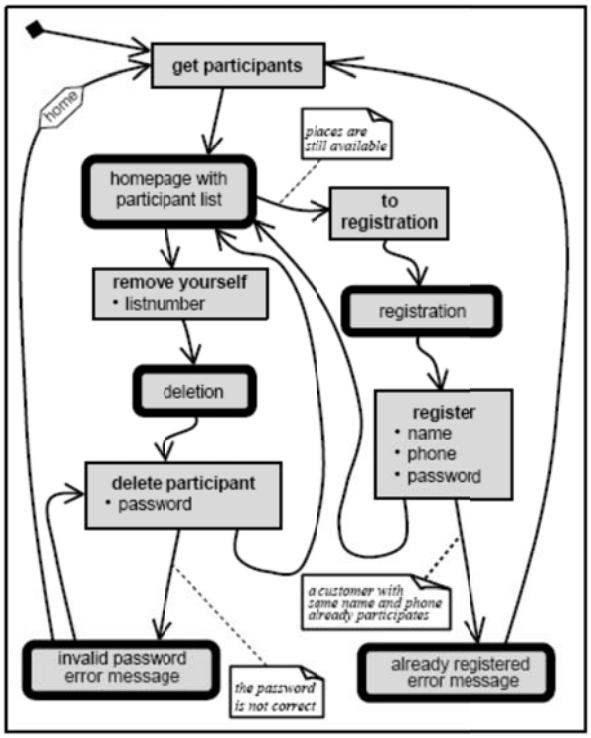
“Un caso de uso es como el argumento general de una película, mientras que el storyboard es como el guión” [49]

Según [49] un storyboard tiene más ventajas que un prototipo regular ya que no depende de una plataforma en específico, además de ser más sencillo de

compartir y no genera falsas expectativas, esto último se debe a que no da la impresión de que el sistema ya está construido.

Una técnica de storyboarding pensada para optimizar la elicitación de requisitos que usa las interfaces basadas en formularios, se muestra en [82]. En la técnica descrita, se toma un formulario como el equivalente a un formulario en papel y se establece la interacción presente en estos sistemas como de tipo envío y respuesta (submit/response). Las propiedades de estos formularios hacen que las transiciones entre las páginas sean más aclarativas en el estudio de la funcionalidad ya que se asume que el estado del sistema en general se representa en los cambios de la interfaz.

Figura 27. Diagrama de transición de estados



En esta técnica se usan los diagramas de transición de estados entre las diferentes pantallas de una interfaz para representar un sistema y se designa como storyboarding de formulario [82]. Un concepto importante, en esta técnica, son las características, estas son pequeñas secciones del diagrama importantes para el análisis ya que contienen o están relacionados con un aspecto especial del sistema.

Una relación con los casos de uso es que estos pueden llegar a definir las rutas en el diagrama de transición de estados o usarse para la identificación de las características. Una de las metáforas de interacción tratadas por esta técnica y que tiene relevancia para nuestro proyecto, es la metáfora de cliente ultra-delgado (ultra-thin client) la cual tiene características similares a las encontradas en el tipo de software heredado de nuestro entorno objetivo (MINUTAS de la DBU - SCC).

La construcción de un storyboard de formulario comienza con la identificación de un número de características inicial, las cuales irán evolucionando a medida que se seleccione y estudie cada una de ellas, es decir se toma cada nodo del diagrama y se evalúa si las interacciones presentes son las requeridas. Las actividades terminaran cuando se halla pasado por todos los nodos del storyboard de formulario, obteniéndose un modelo completo del sistema estudiado.

En conclusión se tiene que el storyboard de formularios no está rígidamente unido a un proceso específico de elicitación de requisitos. Sin embargo ofrece diferentes propuestas para actividades las cuales en sí mismas incrementan la calidad de la especificación de requisitos. Un storyboard de formulario se logra cuando se obtienen todas las características del sistema.

En un nuevo artículo se muestra el uso del prototipado desechable de GUI en la elicitación de requisitos, recurriendo a demostradores de concepto “concept demonstrators” [83]. Esta técnica se basa en facilitarles a los usuarios evaluar el

comportamiento y características de un sistema, a través de una propuesta de interfaz, teniendo en cuenta que los usuarios no se percatan o no pueden comunicar efectivamente las necesidades sin antes haberlas visto en la práctica.

Los primeros prototipos pueden ofrecen formas alternativas para la realización de una tarea, y los usuarios tienen la posibilidad de especificar porque escogen una forma sobre otra. La meta es habilitar al usuario para que de su concepto sobre las alternativas de diseño y en paralelo desarrolle sus propias ideas acerca de la funcionalidad requerida.

Durante la construcción de los prototipos se utiliza un estudio de viabilidad que identifica un conjunto de funciones que guía el diseño del demostrador, definiendo las diferentes alternativas. Para la evaluación de los prototipos y la elicitación de requisitos, se utiliza el método SCRAM (Scenario Requirements Analysis Method) que involucra la presentación de un demostrador conceptual que es ilustrado en una secuencia (scripted sequence) enlazada con escenarios en esta secuencia las posibles funcionalidades y el diseño de interfaces son mostradas. Cada característica del demostrador conceptual es descrita por un facilitador quien pregunta a un pequeño grupo de usuarios acerca de los requisitos.

La etapa final del método envuelve a los usuarios y al facilitador quienes resumen los hechos claves en un mapa de requisitos. Los usuarios realizan un cuestionario previo acerca de la experiencia en el uso del computador y una sesión posterior para proveer de realimentación a las sesiones. En este transcurso se obtienen requisitos funcionales, no funcionales y de usabilidad.

La participación en este tipo de sesiones por parte del usuario fue decisiva y este hecho esta soportado por datos de tiempo y frecuencia los cuales revelan que la demostración de las propuestas animan a los usuarios a exponer sus puntos de vista sin perder la objetividad de la sesión.

Para las siguientes sesiones se pormenorizan los primeros requisitos ya que eran de alto nivel, capturando requisitos de dominios y grupos de usuarios específicos. Para esta técnica fue necesaria una herramienta de prototipado rápido, como una UIDE (user interface development environment) o ambiente de desarrollo de interfaz de usuario, en el caso mostrado en [83] se usó Visual Basic.

El uso de sketches o prototipos de interfaz para obtener un primer modelo conceptual o diagrama de E-R, puede valerse de la interfaz de usuario heredada como especie de prototipo o como una técnica concluyente con construcción del mismo usuario [47].

La siguiente técnica [84] combina la comunicación con los usuarios junto con la referencia a artefactos o construcciones GUI más texto. La meta es el análisis del texto que acompaña la construcción para obtener la información que el usuario quiere transmitir. El enfoque es proveer de una herramienta software que capitalice el “lenguaje de la GUI” combinando la información visual y textual para la facilitación de la comunicación.

La información transmitida únicamente por construcciones graficas puede llegar a ser insuficiente para una adecuada comunicación. Estos artefactos gráficos pueden ser clarificados adjuntando una argumentación textual, permitiendo las fortalezas de ambos tipos de información para aumentar la comunicación.

El colector grafico de requisitos es una herramienta software para la obtención de requisitos directamente de los usuarios finales [85] quienes son instruidos para crear su visión de la interfaz para la aplicación requerida. De esta forma están transmitiendo información de dos formas: Primero la construcción de ventanas y la organización de widgets proveen información a través del lenguaje de la GUI (Fig. n). La argumentación puede ser asociada con los widgets respecto a lo que

representan o realizan, es decir la razón por la cual fueron creados. De esta manera los usuarios tienen un mejor modo de expresar sus propósitos. [85, 86]

6.5 CONCLUSIONES

La ingeniería inversa es la forma en que se aprovecha el software heredado para la realización de nuevo software o en tareas de mantenimiento. El uso del código fuente como recurso para el análisis es costoso en cuanto al procesamiento y por eso han aparecido algunas técnicas que se orientan en las interfaces de usuario, proveyendo a la ingeniería inversa de una fuente para el análisis estático y también dinámico de cualquier sistema sin importar el lenguaje en el que fue construido.

Algunas de las técnicas que se basan en las interfaces se utilizan en la validación de los resultados encontrados, mostrando la información visualmente en lugar de textual, gracias a esto se acelera la identificación de puntos clave y problemas en el proceso de análisis.

7. EVALUACIÓN

La evaluación de las técnicas de obtención de requisitos en la investigación realizada indica que debe ser un proceso iterativo, cuyos ciclos contemplen las técnicas de construcción, abstracción y validación de los requerimientos. El ciclo inicial comienza con una entrevista que pondrá al corriente a todos los interesados de los objetivos globales del proyecto así como de los recursos disponibles. A partir de allí, se analizan los documentos para extraer información del contexto, originar modelos iniciales de casos de uso y de datos y extraer información para poder realizar el análisis del software heredado. La siguiente parte consiste en el análisis del software heredado por las técnicas seleccionadas que perfeccionan los diagramas que se utilizaran en las etapas siguientes del proyecto.

Diagramas como las trazas de la ejecución del sistema y el análisis previo de las pantallas y su comportamiento [72, 47, 78, 71], así como la identificación de la información requerida y los rangos de esta para la correcta manipulación del sistema, brindaran mayor información sobre lo que pasa actualmente en el sistema, para en seguida pasar de los diagramas de secuencia (convertidos en diagramas de procesos) a diagramas de clases [26] y casos de uso.

El enfoque de esta tesis para la comprensión del sistema y dar soporte a la elicitación de requisitos, contempla la interfaz de usuario y sus elementos, sin adentrarnos en el código, ya que la interfaz y sus elementos muestran un nivel más alto de abstracción. La identificación de los componentes de la interfaz, la observación de su comportamiento y la extracción de algunos diagramas se realiza tal como se puede efectuar a través del código, pero usando la grabación de la interacción del usuario con el sistema.

Algunas ventajas del análisis de las interfaces es que no es necesario ser expertos en el lenguaje en el que está hecho el sistema, pues puede tratarse de antiguos lenguajes, sino tener habilidades de reconocimiento de patrones y comportamientos, observables en las interfaces y que además se puede abarcar un sistema bastante complejo, en términos del código, sin demasiado esfuerzo.

Un cuadro que examina algunas de las dificultades presentes en el análisis de las interfaces, se muestra a continuación:

Tabla 4. Características

Característica	Desventajas
Herramientas	Necesita instrumentación (En comparación con las herramientas, más complejas y que consumen más tiempo, necesarias para el análisis estático del código relacionado con la interfaz, no es una desventaja, en resumen la técnica presenta el análisis de la interfaz como una buena fuente de información y establece algunos criterios para automatizaciones posteriores).
Restricción de acciones	Las acciones de borrado de la BD se requieren para observar la función (Esto no es tanto una desventaja ya que para la obtención de trazas se interacción se observarán las operaciones propias del negocio). Necesita del enfoque de prueba y error (Como ya se dijo se trabaja con las operaciones propias de los usuarios lo que indica el mejor camino posible o el flujo normal de la actividad).
Completitud	Se puede pasar por alto alguna pantalla (Las trazas de ejecución serán lo más completas posibles, se espera que no falte ninguna pantalla, a menos que no sea utilizada, lo que indicaría comparado con el análisis previo de toda la interfaz, que es un recurso subutilizado).
Código	Los resultados no serán enlazados con las porciones de código relevantes (El objetivo es realizar una especificación del sistema en un nivel más alto de abstracción y que en su mayoría no dependa de la plataforma).

A continuación se sigue con las entregas de prototipos luego de la segunda iteración, esto para la validación de los requisitos obtenidos, en donde se espera otra realimentación de parte de los usuarios.

Para los prototipos nuestro enfoque se orienta hacia los prototipos de interfaz, es decir más parecidos a sketch, que además de validar la secuencia de ventanas y correspondientes acciones para la realizar las tareas contenidas en los requisitos, tiene la posibilidad de mostrar el nuevo entorno (por ejemplo el entorno Web) que tendría una aplicación. En cuanto a su desarrollo se proyecta que sean prototipos desechables por las características de rapidez y sencillez entre otras.

También y debido a la proliferación de estilos de diseño es posible que una vez sea acordado el estilo por parte de analistas y usuarios, se utilice un meta-esquema [36] para los prototipos de interfaces, esto con el fin de estandarizar la generación de prototipos y seguir con el análisis más enfocado en las funcionalidades.

La tarea de elicitación, sufre tres tipos de problemas [23]:

- Dificultades en la definición del alcance
- Dificultades en la comprensión entre los analistas e interesados
- Dificultades de volatilidad

Los cuales se superan gracias a que la definición del alcance se realiza de acuerdo a la realimentación de la etapa de elicitación, basándonos en los prototipos lo que aumenta el grado de entendimiento entre los participantes en la evaluación.

Y ya que nos estamos basando en el software heredado, se tiene una base sólida y conocida para la definición del sistema requerido.

Las **herramientas** que se tienen en cuenta para apoyar estas técnicas de observación y especificación de prototipos de interfaz son variadas y dependen de los aspectos de las técnicas a las que soportan.

La herramienta de observación Recorderse utiliza en la técnicaCelLEST y consiste en que un analista graba e interpreta trazas de la interacción del usuario con el sistema, estas trazas corresponden a escenarios propios de la operación del negocio.

La herramienta de LENDI (LEgacy Navigation Domain Identifier)identifica y agrupa las pantallas recopiladas por el Recorder obteniéndose un gráfico dirigido llamado gráfico de interfaz.

La herramienta URGENT (User interface ReGENeration Tool)lleva a cabola siguiente tarea de modelado de tareas y el dominio.

Las trazas son la entrada para otra actividad la cual las analiza y obtiene patrones de funcionalidades que son luego convertidas en casos de uso. Las trazas tomadas a partir de los diferentes escenarios son agrupadas para conseguir una coherencia conceptual con la operación del negocio. Cada traza se estudia para obtener patrones que podemos utilizar como flujos principales o alternativos en los casos de uso.

En esta etapa algunos problemas del software heredado son encontrados, teniéndose que comenzar a realizar decisiones de reparación. Los casos de uso iniciales (encontrados solo a partir de las trazas) evolucionaran para solventar algunas deficiencias, lográndose una mejora de la especificación original.

Una vez obtenidos los casos de uso, se utilizan los diagramas de secuencia del sistema para representar más información contenida en las trazas. Un diagrama

de secuencia es obtenido por cada caso de uso encontrado y contendrá los objetos de información y actores que maneja cada uno de ellos.

Los diagramas de secuencia del sistema serán la base para completar el modelo conceptual del dominio del problema (MCDP), inicialmente previsto en la técnica de análisis de documentos, encontrando los objetos del negocio y las relaciones que se muestran en la funcionalidad actual del sistema. Puede haber dos situaciones en el manejo de los objetos y relaciones del MCDP, encontrarse con una esquema de datos fijo o tener completa libertad sobre las decisiones del MCDP.

La primera circunstancia permite un movimiento limitado para las propuestas de objetos y relaciones ya que debe corresponder con el esquema definido en el sistema. Las propuestas se realizan en base a lo encontrado de la funcionalidad mejorando aspectos de navegación y usabilidad, así como presentación y contenido. En la situación en la cual hay libertad sobre las decisiones de diseño los diagramas pueden llegar a representar los intereses del analista, sin dejar de lado las funcionalidades observadas en el software heredado.

Para una tarea de elicitación de requisitos sin inconsistencias es necesaria la participación (aunque según lo expuesto, solo en la medida precisa) de los usuarios que validan lo obtenido, con este fin se utilizan los prototipos de interfaz. Estos prototipos realizan una demostración del modelo de requisitos, obtenido por las técnicas anteriores, pero de forma gráfica y más cercana con el usuario, permitiéndole ver navegación y funcionalidad (estática) e estimulándolo para que participe con su experiencia y conocimiento en la completitud de los requisitos.

Las herramientas utilizadas son las que permitan realizar los diferentes estilos y niveles de prototipos, desde tableros, hasta aplicaciones de dibujo orobustos ambientes de diseño.

8. RECOMENDACIONES

En [87] se presenta la Identificación de requisitos en el desarrollo de software basado en componentes, utilizando repositorios de componentes CKB (Component Knowledge-Base), la identificación de los componentes para esta tecnología tiene en cuenta que los componentes son código más una interfaz que ejecuta un servicio, es así como el paradigma de “reconocer lo ya visto”, en este caso la interfaz gráfica, puede ser útil para identificar lo que se desea de un sistema nuevo. Además podemos hablar de que estos componentes son un tipo de software heredado, al cual utilizamos sin modificar pero que de manera integrada nos genera un nuevo sistema.

BIBLIOGRAFÍA

[1] Escudero, Consuelo. Moreira, Marco Antonio. La V epistemológica aplicada a algunos enfoques en resolución de problemas. Enseñanza de las ciencias. 1999, Pág. 66.

[2] Tomayko, James. Engineering of Unstable Requirements Using Agile Methods.IEEE. 2002.

[3] Horrian, Hossein. Mahmud, Shafquat. Karthikeyan, Srinivasan. Requirements Engineering in Agile methods.Departamento de ciencias de la computación, Universidad Calgary, Canada, 2003.

[4] Vilalta, Josep. La “Hoja de Ruta” UML. Estrategias docentes de UML y MDA. 2004.

[5] Vilalta, Josep. Que Es UML. Estrategias docentes de UML y MDA. 2004.

[6] Young, Ralph R. The Requirements Engineering Handbook. ARTECH HOUSE, INC. 2004.

[7] Ingeniería de Requisitos. Material de Ingeniería del Software I. Universidad Rey Juan Carlos.

[8] Requirements Engineering. EDS Company. 2007

[9] DURÁN, Amador. Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Departamento de Lenguajes y Sistemas InformáticosUniversidad de Sevilla. 2000.

[10] Quintero, Juan Bernardo. Requisitos de Software. Presentación de diapositivas.

[11] Sommerville, Ian. Software Engineering.6th ed. 2000.

[12] Choudhary, Adnan. Requirement Engineering – A Roadmap.

[13] Rajlich, Vaclav. Comprehension and evolution of legacy software. Departamento de Ciencias de la Computación, Universidad del Estado Wayne.

[14] IEEE 610.12_1990 - Standard Glossary of Software Engineering Terminology

[15] Business Rules Hidden in Use Cases, Articulo Web Tyner Blain, 2007

[16] Borysowich, Craig. Overview of Jacobson's Use Case Approach.Articulo WebITtoolbox Blogs.2007

[17] Durán Toro, Amador. Bernárdez Jiménez, Beatriz. Metodología para la Elicitación de Requisitos de Sistemas Software - Versión 2.3. Universidad de Sevilla. 2002.

[18] Williams, Laurie. Requirements Engineering and Elicitation. Requirements engineering. 2004

[19] Kotonya, G. Sommerville, Ian. Requirements Engineering: Processes andTechniques. 1998.

[20] ESPECIFICACIONES DE LOS REQUISITOS DEL SOFTWARE. IEEE-STD-830-1998.

[22] Cockburn, Alistair. Basic Use Case Template.ACM. 1998.

[23] A. Durán Toro, B. Bernárdez Jiménez, A. Ruiz Cortés, and M. Toro Bonilla. A Requirements Elicitation Approach Based in Templates and Patterns. Departamento de Lenguajes y Sistemas Informáticos. Facultad de Informática y Estadística. Universidad de Sevilla.

[24] Ceria, Santiago. Casos de Uso – Un Método Práctico para Explorar Requerimientos. Ingeniería de Software I.

[25] Use Case Model. Articulo Web Enterprise Architect.

[26] García Molina, Jesús. Ortín, M. José. Moros, Begoña. Nicolás, Joaquín.Toval, Ambrosio. De los Procesos del Negocio a los Casos de Uso. WebSite: http://www.cyta.com.ar

[27] English, Arthur.Business modeling with UML Understanding the similarities and differences between business use cases and system use cases.Articulo Web IBM. 2007.

[28] Ten Requirements Gathering Techniques. Articulo Web Tyner Blain. 2006

[29] Yu, Yijun.Mylopoulos, John. Et al.RETR: reverse engineering to requirements. Working Conference on Reverse Engineering.IEEE Computer Society. 2005.

[30] Ibrahim, Rosziati. Yong, Tiu Kian.ReSeT: Reverse Engineering System Requirements Tool. Proceedings of World Academy of Science, Engineering and Technology.Vol. 32.2008.

[31] Carrizo Moreno, Dante. Caracterización de las Técnicas de Adquisición de Requisitos. WebSite: http://www.dlsiis.fi.upm.es/docto_lsiis/Trabajos20012002/DCarrizo.doc

[32] El-Ramly, Mohammad. Stroulia, Eleni. Sorenson, Paul. From run-time behavior to usage scenarios: an interaction-pattern mining approach. Department of Computing Science University of Alberta.ACM. 2002.

[33] IEEE 1233_1998 - Guide for Developing System Requirements Specifications. Software Engineering Standards Committeeof theIEEE Computer Society. 1998.

[34] Canfora, Gerardo. Di Penta, Massimiliano. New Frontiers of Reverse Engineering. Future of Software Engineering. IEEE Computer Society. 2007.

[35] Weide, Bruce. Heym, Wayne. Reverse engineering of legacy code exposed. Department of Computer and Information ScienceThe Ohio State University. ACM. 1995

[36] Sneed, Harry. Sneed,Stephan. Reverse Engineering of System Interfaces A Report from the Field. Working Conference on Reverse Engineering.IEEE Computer Society. 2006.

[37] Moore, Melody.Reverse Engineering User Interfaces- A Technique.College of ComputingGeorgia Institute of Technology.

[38] Ibrahim, Rosziati. Yong, Tiu Kian.ReSeT: Reverse Engineering System Requirements Tool. Proceedings of World Academy of Science, Engineering and Technology.Vol. 32. 2008.

[39] Staiger, Stefan.Reverse Engineering of Graphical User Interfaces Using Static Analyses.Institute of Software TechnologyUniversity of Stuttgart.IEEE Computer Society.

[40] SIOCHI, ANTONlO C. EHRICH, ROGER W. Computer analysis of User interfaces Based on Repetition in Transcripts of User Sessions. ACM Transactions on Information Systems.Vol. 9. 1991.

[41] Stroulia, Eleni. El-Ramly, Mohammad. Kong, L. Sorenson, Paul. Matichuk, B. Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach. Proceedings of the 6th Working Conference on Reverse Engineering.IEEE Computer Society. 1999.

[42] El-Ramly, Mohammad. Stroulia, Eleni. Sorenson, Paul. Mining System-User Interaction Traces for Use Case Models.Proceedings of the 10 th International Workshop on Program Comprehension.IEEE Computer Society. 2002.

[43] Lloyd,Wes. Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study. Faculty of theVirginia Polytechnic Institute and State University. 2001.

[44] Hickey, Ann. Davis,Alan. Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. Hawaii International Conference on System Sciences.IEEE Computer Society. 2002.

[45] Christel, Michael. Kang, Kyo.Issues in Requirements Elicitation.Software Engineering InstituteCarnegie Mellon University. 1992.

[46] Nuseibeh, Bashar. Easterbrook, Steve. Requirements Engineering: A Roadmap. Future of Software engineering Limerick Ireland.ACM. 2000.

[47] Brogneaux, Anne-France. Ramdoyal, Ravi. Vilz, Julien. Hainaut, Jean-Luc.Deriving User-requirements From Human-Computer Interfaces.Laboratory of Database Application Engineering.University of Namur.

[48] Landay, James. Myers,Brad. Sketching Storyboards to Illustrate Interface Behaviors. HCI Institute, School of Computer Science Carnegie Mellon University.

[49] Krebs, Jochen. Form feeds function: The role of storyboards in requirements elicitation. Articulo IBM. 2005.

[50] Luedke, Betty. Requirements gathering with storyboards. Articulo TechTarget. 2008.

[51] Granollers i Saltiveri. Toni. MPIu+a. Una Metodología queIntegra la Ingeniería del Software,la Interacción Persona-Ordenadory la Accesibilidad en el Contextode Equipos de Desarrollo Multidisciplinares.Universidad de Lleida. 2004

[52] Pichler, Mario. Rumetshofer, Hildegard. Business Process-based Requirements Modeling and Management.Workshop on Requirements Engineering Visualization.IEEE Computer Society. 2006

[53] Lloyd, Wes. Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study. Tesis de Maestría para el Instituto Politécnico y Universidad Estatal de Virginia.2001.

[54] Leffingwell, D. Widrig, D. Managing Software Requirements: a Unified Approach. Addison-Wesley Longman Publishing Co., Inc. 2000.

[55] Maner, Walter. Prototipado.Articulo Web Sid@r. WebSite: http://www.sidar.org/recur/desdi/traduc/es/visitable/Otros.htm

[57] Olsina, Luis.Building a Web-based information system applying the hypermedia flexible process modeling strategy.International Workshop on Hypermedia Development.

[58] Lowe,David. Client Needs and the Design Process in Web Projects.Journal of Web Engineering, Vol. 1.Rinton Press. 2002.

[59] Escalona, María José. Koch, Nora. Ingeniería de Requisitos en Aplicaciones para la Web:Un estudio comparativo.

[61] Van Lamsweerde, Axel.Willemet, Laurent.Inferring Declarative Requirements Specifications from Operational Scenarios.IEEE Transactions on Software Engineering, Vol. 24.No. 12. 998.

[62] Moore, Melody. MORALE METHODOLOGY GUIDEBOOK – Model Oriented Reengineering Process for Human-Computer Interface(MORPH). College of ComputingGeorgia Institute of Technology.

[63] Moore, Melody. Glosario de Términos de Interacción.1996. Sitio: http://www.cc.gatech.edu/morale/local/morph_glossary.html

[64] Imagen obtenida de internet del sitio: **http://trac.osgeo.org/grass/attachment/wiki/HowToTestGrass6/grass64_text_start.png,** 10 de octubre de 2010.

[65] Stroulia, Eleni. El-Ramly, Mohammad. Kong, L. Sorenson, Paul. Matichuk, B. Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach. Working Conference on Reverse Engineering. IEEE Computer Society. 1999.

[66] Imagen obtenida de internet del sitio: **http://i4.tinypic.com/86ij14z.jpg.** 10 de octubre de 2010

[67] Macias, José. Puerta, Ángel. Castells, Pablo. MBUI para Procesos de Ingeniería Inversa. Interacción'04. Lleida, España. 2004

[68] Leite, Jair C. A semiotic-based framework to user interface design. NordiCHI 9 Århus, Denmark, 2002 ACM

[69] VanderDonckt, Jean M. Bodart, Francois. Encapsulating knowledge for intelligent automatic interaction objects selection. ACM Conference on Human Aspects in Computing Systems InterCHI. 1993.

[70] Molina M., Pedro. Especificación de Interfaz de Usuario: De los requisitos a la generación automática. Tesis doctoral. Universidad politécnica de Valencia. p 23. 2003.

[71] Molina Moreno, Pedro Juan. Torres Boigues, Ismael. Pastor López, Oscar. Human-Computer Interaction: Overcoming Barriers - User Interface Patterns for Object-Oriented Navigation. UPGRADE The European Online Magazine for the IT Professional. Vol. IV, No. 1, p. 31, 2003

[72] Choobineh, Joobin. Mannino, Michael. Tseng, Veronica. A Form-Based Approach for Database Analysis and Design. Communications of the ACM. 1992.

[73] Lee, Heseeok. Yoo, Cheonsoo. A form driven object-oriented reverse engineering methodology. Elsevier Science Ltda. 2000.

[75] Stroulia, Eleni. El-Ramly, Mohammad. Kong, L. Sorenson, Paul. Matichuk, B. Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach. Proceedings of the 6th Working Conference on Reverse Engineering. IEEE Computer Society. 1999.

[76] Zaidma, Andy. Calders, Toon. Demeyer, Serge. Paredaens, Jan. Applying Webmining Techniques to Execution Traces to Support the Program Comprehension Process. University of Antwerp Department of Mathematics and Computer Science.

[77] El-Ramly, Mohammad. Stroulia, Eleni. Sorenson, Paul. Recovering Software Requirements from System-user Interaction Traces. Department of Computing Science, University of Alberta. 2002

[78] Insfrán, Emilio. Molina, Pedro J. Sofía, Martí. Pelechano, Vicente. Ingeniería de Requisitos aplicada al modelado conceptual de interfaz de usuario. IDEAS 2001. Pag. 181-192. Santo domingo, Costa Rica. 2001.

[79] English. Arthur. Business modeling with UML Understanding the similarities and differences between business use cases and system use cases. IBM. 2007

[80] Martinez, Alicia. Estrada, Hugo. Sanchez, Juan. Pastor, Oscar. From early requirements to user interface prototyping: a methodological approach. Proceedings of the 17 th IEEE International Conference on Automated Software Engineering. IEEE Computer Society. 2002.

[81] Coyette, Adrien. Kieffer, Suzanne. Vanderdonckt, Jean. Multi-fidelity Prototyping of User Interfaces. INTERACT 2007. IFIP International Federation for Information Processing 2007. Pag. 150–164, 2007.

[82] Draheim, Dirk. Weber, Gerald. An Introduction to Form Storyboarding.Reportetecnico. 2002.

[83] Ryan, Michele. Doubleday, Ann. Evaluating 'Throw Away' Prototyping for Requirements Capture. Centre for HCI Design, School of Informatics

[84] Moore, Michael. Shipman, Frank M. Requirements Elicitation using Visual and Textual Information.IEEE. 2001

[85] Moore, Michael. Shipman, Frank M. A Comparison of Questionnaire-Based and GUI-Based Requirements. Department of Computer Science Texas A&M University.

[86] Moore, Michael. Communicating Requirements Using End-User GUI Constructions with Argumentation. Department of Computer Science Texas A&M University.

[87] Jain, Hemant. Vitharana, Padmal. Zahedi, Fatemah. An Assessment Model for Requirements Identification in Component-Based Software Development. The DATA BASE for Advances in Information Systems. 2003

[88] Mezquita, Joel (2006). Tesis Doctoral: Recuperación de Información con resolución de ambigüedad de sentidos de palabras para el español.

[89] M. Maron. Automatic indexing: an experimental inquiry. Journal of the ACM, 8:404–417, 1961. C.J. van Rijsbergen. Information retrieval. 1979

[90] G. Salton and M. E. Lesk, “Computer Evaluation of Indexing and Text Processing,” Journal of the ACM, 15(1):8-36, January 1968.

[91] G. Salton, “The SMART Retrieval System - Experiments in Automatic Document Processing,” Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[92] L. Liu. “Undestanding how the Requirements are Implemented in Source Code”. Software Engineering Institute. Pekin University.

[93] Galicia-Haro Sofía N., Bolshakov I. A. y Gelbukh A. F.” Un modelo de descripción de la estructura de las valencias de verbos españoles para el análisis automático de textos”. 1999.

[94] Zapata Mario “Los modelos verbales en el lenguaje natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software”. Universidad Eafit. 2006.

[95] Buchholz. ”Applyning a Natural Languahe Dialogue Tool For Designing Data-Bases”. En: Proceedings of the Fisrt International Workshop on Applications of Natural Language to Databases. 1995. P.15.

[96] Gervasi. “The CICO domain-basedparser”. Reporte técnico TR-01-25, Departamento de Informática, Universidad de Pisa. 2001. P. 52

[97] Gervasi. Environment support for requirements writing and analysis. PhD Thesis, Universidad de Pisa. 1999.

[98] Ambriola y Gervasi. “On the parallel refinement of NL requirements an UML diagrams”. En: Proceedings of the ETAPS 2001 Workshop on Transformatios in UML, Genova. 2001. P 5.

[99] Cyre, W. “A Requirements sublanguage for automated analysis”. En: International Journal of Intelligent Systems”. 1995. Vol 10. P. 665-689.

[100] Fillmore. “The case for Case”. En: Universals in Linguistics, Bach and Harms, editors. 1968. p. 1-88.

[101] Mich. “NL-OOPS: From Natural Language to Object Oriented Requirements using the Natural Language processing System LOLITA”. En journal of Natural Language Engineering, Cambridge University Press, 1996. Vol. 2, No.2. pp. 161-187

[102] Mich y Garigliano. “NL-OOPS: A Requirements Analysis tool basedon Natural Language Processing”. En: Proceedings of the 3rd International Conference on Data Mining 2002, Bologna. Pp. 321-330.

[103] Dardenne. “Goal-Directed requirements acquisition”.En Science of computes programming, Vol 20. 1993. P 3-50.

[104] Van Lansweerde y letier . “Integrating obstacles in goal- driven requirements engineering”. En: proceedings 20th conference on software engineering, Kioto.1998 P 10.

[105] Codina, Lluís. “Teoría de recuperación de información: modelos fundamentales y aplicaciones a la gestión documental”. Artículo on-line: < http://www.elprofesionaldelainformacion.com/contenidos/1995/octubre/teora_de_re

cuperacin_de_informacin_modelos_fundamentales_y_aplicaciones_a_la_gestin_documental.html> Visitado el 15 de Octubre de 2010.