TÉCNICA PARA LA ADQUISICIÓN DE REQUERIMIENTOS DE SOFTWARE MEDIANTE INGENIERÍA INVERSA APLICADA A INTERFACES GRÁFICAS DE USUARIO DE SISTEMAS HEREDADOS

CAROL LISET JAIMES VEGA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2016

TÉCNICA PARA LA ADQUISICIÓN DE REQUERIMIENTOS DE SOFTWARE MEDIANTE INGENIERÍA INVERSA APLICADA A INTERFACES GRÁFICAS DE USUARIO DE SISTEMAS HEREDADOS

CAROL LISET JAIMES VEGA

Trabajo de investigación presentado como requisito parcial para optar al título de:
Magister en Ingeniería de Sistemas e Informática

Director:
MSc. Fernando Antonio Rojas Morales
Maestría en Informática

Ingeniería del Software
Grupo de Investigación:
Grupo de Investigación en Ingeniería Biomédica GIIB

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2016

Este trabajo está dedicado muy especialmente:

A mi madre Dora por su cariño y apoyo incondicional en este importante reto, el cual sin ella no hubiera sido posible.

A mi padre Robiel por estar siempre conmigo, ser la inspiración y mi ángel guardián, quien me llevo por esta senda y al que debo en gran medida todo lo que soy.

A Victor Manuel por su amor, paciencia, comprensión y uno que otro sermón en los tiempos difíciles.

Y por último y no menos importante a mi mejor amiga Lorena quien me ha aguantado el genio y los sarcasmos desde antes de escribir dedicatorias como esta.

AGRADECIMIENTOS

Primero agradezco a Dios por la vida, la salud y las oportunidades dadas. También quiero expresar mis más infinitas gracias a:

El profesor FERNANDO ANTONIO ROJAS MORALES por el apoyo, la orientación y la confianza puesta en mí desde que iniciamos este proceso.

Al ingeniero HECTOR CAMILO ZAMBRANO HERNANDEZ director operativo de MTI S.A.S por brindarme la oportunidad desarrollar un proyecto aplicable al ámbito técnico real y por facilitarme los recursos necesarios para su consecución.

A la UNIVERSIDAD INDUSTRIAL DE SANTANDER, por forjarme como una profesional e investigadora íntegra, con alto sentido de la responsabilidad y siguiendo siempre altos estándares de calidad.

Y a todos mis AMIGOS y COMPAÑEROS quienes en cada momento dado brindaron su aporte, palabra de alivio y apoyo para la consecución de este logro.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

ANÁLISIS DINÁMICO: el Análisis dinámico de software es un tipo de análisis de software usado en ingeniería inversa que supone la ejecución del programa y observar su comportamiento. Para que el análisis dinámico resulte efectivo el programa a ser analizado se debe ejecutar con los suficientes casos de prueba como para producir un comportamiento interesante.

ANÁLISIS ESTÁTICO: es un tipo de análisis de software utilizado en ingeniería inversa que se realiza sin ejecutar el programa (el análisis realizado sobre los programas en ejecución se conoce como análisis dinámico de software). En la mayoría de los casos, el análisis se realiza en alguna versión del código fuente y en otros casos se realiza en el código objeto. El término se aplica generalmente a los análisis realizados por una herramienta automática, el análisis realizado por un humano es llamado comprensión de programas (o entendimiento de programas) como también revisión de código.

ÁRBOL DE INTERFACES: Representación gráfica del listado de interfaces propuesto en la técnica ReFree, este permite visualizar la forma en que es posible navegar a través de las interfaces de la aplicación analizada.

ARQUITECTURA DE LA INFORMACIÓN (AI): la cual es el arte de organizar un espacio de información, lo cual abarca la presentación, búsqueda, navegación, etiquetado, categorización, organización, manipulación y encubrimiento estratégico de la información.

CARACTERÍSTICAS: Una característica es una descripción de alto nivel de algo que el sistema necesita hacer. Las características se obtienen generalmente mediante interacciones con los clientes.

CICLO DE VIDA DE SOFTWARE: algunas veces usado como otro nombre para el proceso de software; originalmente este término fue acuñado para referirse al modelo en cascada del proceso de software.

DESARROLLO ORIENTADO A CARACTERÍSTICAS: el Desarrollo Orientado a las Características (Feature Driven Development, FDD) es una metodología de software ágil y adaptativo que busca diseñar y probar con el fin de entregar resultados de trabajo frecuentemente. El FDD es un proceso altamente iterativo, se enfoca en la calidad en cada paso, entrega resultados de trabajo tangibles y provee información exacta y significativa sobre el estatus del proyecto.

FUENTES DE REQUERIMIENTOS: literatura, personas, cosas, procesos o sistemas de los cuales se extraen los requerimientos de un sistema de software.
INGENIERÍA DE SOFTWARE: (1) la aplicación de un enfoque sistemático, disciplinado y cuantificable del desarrollo, operación y mantenimiento de software,

esto es la aplicación de ingeniería al software y (2) El estudio de los enfoques en (1).

INGENIERÍA DIRECTA: proceso de ingeniería tradicional el cual inicia con el diseño para proceder a la construcción, pruebas y posterior mantenimiento de un artefacto.

INGENIERÍA INVERSA: proceso de extraer el conocimiento o modelo del diseño de cualquier cosa hecha por el hombre.

INGENIERÍA INVERSA DE SOFTWARE: la ingeniería inversa es el proceso de analizar artefactos de software disponibles, tales como requerimientos, diseño, arquitecturas, código y código byte, con el objetivo de extraer información y proveer vistas de alto nivel del sistema en cuestión

INTERESADOS: los involucrados o interesados (stakeholders en inglés) son todas aquellas personas u organizaciones que son afectados o pueden tener alguna influencia sobre el proyecto de desarrollo de software.

INTERFAZ DE USUARIO: la interfaz de usuario es el mecanismo mediante el cual el usuario se comunica con el computador, es decir provee tanto el mecanismo de entrada, donde el usuario le “dice” al computador qué hacer, como el mecanismo de salida, a través del cual el computador responde a usuario. Las personas interactúan con un computador a través de interfaz usando el teclado, ratón, pantallas táctiles y micrófonos.

INTERFAZ GRÁFICA DE USUARIO: una interfaz gráfica de usuario, o GUI, es la evolución de la interfaz que incluye elementos reusables de la UI y es soportada por dispositivos móviles, la cual no necesariamente usa puntero de ratón. Esta permite interactuar con la aplicación usando gráficos, imágenes, iconos, y elementos en la pantalla 2-D, sin tener que memorizar complejos comandos y digitarlos de manera precisa usando el teclado, tal como en la CLI.

LISTADO DE INTERFACES: el listado de interfaces es el inventario de todas las interfaces que van a ser analizadas por medio de la técnica ReFree, este se representa por medio de una lista ordenada en la forma de navegación de la aplicación. Gráficamente se representa con el árbol de interfaces.

MATRIZ DE COMPONENTES: registro ordenado de la descomposición en componentes gráficos de cada una de las interfaces que van a ser sometidas a análisis por medio de ReFree.

MODELO INTERACCIÓN: Una aplicación tiene una o múltiples páginas para realizar las tareas que le permitan cumplir con dichos servicios. Cualquier página que haga parte de una aplicación realizará primariamente una de las siguientes

cosas: Mostrar una sola cosa, mostrar una lista o conjunto de cosas, proveer herramientas para crear una cosa o facilitar una tarea. Lo anterior se consideran modelos de interacción y establecen consistencia a través del artefacto, y determina como los usuarios se mueven a través y entre las diferentes piezas de funcionalidad.

MODELO NAVEGACIÓN: los modelos de navegación son la forma en la que las pantallas (páginas o espacios) se unen unas a otras, y como los usuarios se mueven a través de ellas en un sistema de software.

ONTOLOGÍA: Una ontología es una descripción explícita y formal de los conceptos en un dominio de discurso. Se conforma de clases (algunas veces llamadas conceptos), propiedades de cada concepto que describen diferentes características y atributos de los conceptos (estas propiedades son llamadas ranuras, roles o simplemente propiedades); y también consta de restricciones sobre las ranuras o facetas (llamadas restricciones de rol). Una ontología junto con un conjunto de instancias de las clases constituye una base de conocimiento.

PARÁMETROS INICIALES: características de un sistema heredado registradas en la primera etapa de ReFree y generalmente obtenidas por simple observación o una entrevista breve con algún usuario de la aplicación a analizar.

PROCESO DE SOFTWARE: conjunto de actividades relacionadas y procesos que están involucradas en el desarrollo y evolución de un sistema de software.

REFREE: Sigla para Reverse Engineering For Requirements Elicitation, técnica para la captura de requerimientos de software a partir del análisis de la interfaz gráfica de usuario de un sistema heredado.

REINGENIERÍA: conocida también como renovación y reclamación, es la examinación y alteración de un sistema terminado para reconstituirlo en una nueva forma y la subsecuente implementación de la nueva forma. Involucra la ingeniería directa y la inversa pero no es un super tipo de las dos.

REQUERIMIENTO: enunciado de alguna función o característica que debe ser implementada en un sistema.

SISTEMA HEREDADO: sistema socio – técnico que es útil o esencial en una organización, pero que fue desarrollado usando tecnología o métodos obsoletos. Debido a que los sistemas heredados desarrollan operaciones críticas para las organizaciones, estos deben someterse a importantes actividades de mantenimiento.

RESUMEN

TÍTULO: TÉCNICA PARA LA ADQUISICIÓN DE REQUERIMIENTOS DE SOFTWARE MEDIANTE INGENIERÍA INVERSA APLICADA A INTERFACES GRÁFICAS DE USUARIO DE SISTEMAS HEREDADOS[1]

AUTOR: Carol Liset Jaimes Vega[2]

PALABRAS CLAVE: Requerimientos, ingeniería de software, ingeniería inversa, GUI, características, lenguaje natural.

DESCRIPCIÓN:

La ingeniería de software tiene cuatro etapas fundamentales: Requerimientos, Diseño, Construcción, Pruebas y Mantenimiento. La primera actividad relacionada con la etapa de Requerimientos es la Captura de Requerimientos, esta es una actividad obligatoria cuando se enfrenta un nuevo desarrollo de software. La Captura de Requerimientos es una etapa extremadamente importante pero en la mayoría de los casos se relaciona con una actividad de "segunda mano", sin tiempo, sin recursos e incluso sin personal adecuado. La importancia de la Captura de Requerimientos está dada por el alto costo y dificultad de corregir los errores que se comentan durante su realización y que son evidentes en fases posteriores del desarrollo.

Este trabajo presenta el desarrollo de una técnica para la Captura de Requerimientos a partir de sistemas heredados aplicando Ingeniería Inversa a sus Interfaces Gráficas de Usuario (GUIs). La técnica se aleja de otros enfoques de Ingeniería Inversa como el clásico análisis estático y dinámico del código. En lugar de esto la técnica busca dentro de los componentes de la GUI y sus patrones de diseño, respuestas sobre la funcionalidad de la aplicación heredada.

La técnica propuesta usa la extracción del lenguaje de la GUI para recuperar descripciones de las funcionalidades del sistema heredado. También organiza las funcionalidades capturadas en características, que son un formato estándar de requerimientos, dando al diseñador la oportunidad de usar estos requerimientos capturados en un nuevo proceso de desarrollo.

[1] Trabajo de Investigación

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Fernando Antonio Rojas Morales, Magíster en Informática.

ABTRACT

TITLE: REQUIREMENTS ELICITATION TECHNIQUE USING REVERSE ENGINEERING OF LEGACY SYSTEMS GRAPHICAL USER INTERFACES[3]

AUTHOR: Carol Liset Jaimes Vega[4]

KEYWORDS: Requirements, software engineering, reverse engineering, GUI, features, natural language.

DESCRIPTION:

Software Engineering has four stages: Software Requirements, Design, Construction, Testing and Maintenance. The first activity related with Software Requirements is the Requirements Elicitation; this is a "must" activity when a new development has to be faced off. Requirements Elicitation is an extremely important stage but in the most of cases it is related with a second-hand activity, with no time, no resources and even no qualified personal designated. The importance of Requirements Elicitation is given by the high cost and difficulty to correct errors committed during this stage that appear in posterior project phases.

This work presents the technique development for requirements elicitation from legacy systems applying Reverse Engineering to their Graphic user Interfaces (GUIs). The technique tries to be quite far from other reverse engineering approaches like the classic static and dynamic code analysis, and instead of this, it searches inside the GUIs components and design patterns answers about the legacy application functionality.

The technique uses GUIs language extraction to recover descriptions of the legacy software functionalities. The technique also organizes the captured functionalities in features, which is a standard requirements format, giving to the designer the opportunity to use this captured requirements in a new development process.

[3] Master's degree research work.
[4] Departament of Physical-mechanical Engineering. School of Systems Engineering and Computer Science. Advisor: Fernando Antonio Rojas Morales, M. Sc. Computer Science.

INTRODUCCIÓN

En el contexto de la ingeniería de software, la captura de requerimientos se considera una actividad esencial, que produce el insumo para el resto del proceso de software. A pesar de que la mayoría de metodologías proponen actividades de intensa interacción con el usuario para lograr una captura de requerimientos de calidad, la realidad es que el usuario no dispone del tiempo necesario. Si además, se toma en cuenta el escenario actual de desarrollo de software, es un hecho que en la mayoría de organizaciones se cuenta con aplicaciones. Lo que da origen a la oportunidad de realizar una de las actividades más importante dentro de la captura de requerimientos: la adquisición del conocimiento del dominio, infiriéndolo desde dichas aplicaciones (conocidas como software heredado).

En este trabajo de investigación se busca dar una nueva perspectiva a la Captura de Requerimientos y a las hasta ahora fuentes identificadas y usadas para recolectar estos requerimientos. Se presenta una alternativa que basada en enfoques cásicos permite explotar los sistemas heredados y a través de estos reconocer elementos de funcionalidad relacionados con el dominio de la aplicación.

Para lograr esto se aborda la interfaz gráfica de los sistemas heredados a partir de la simple observación y reconocimiento del lenguaje implícito que esta contiene. Seguidamente se propone una traducción a un formato de requerimientos basado en lenguaje natural y se realizan pruebas en aplicaciones software que difieren tanto en su tipo como en su dominio.

Se presenta a continuación todo el desarrollo conceptual y metodológico requerido para llevar a cabo el presente proyecto de investigación, así como también los resultados y las conclusiones a las cuales se llegó a través de la experiencia y cómo estás demuestran la validez de las cuestiones planteadas en el planteamiento del problema.

1. PLANTEAMIENTO DEL PROBLEMA

La implementación de aplicaciones de software ha pasado a través de los años de ser vista como una actividad empírica a tener un fundamento en procesos ingenieriles claros y sistemáticos. De manera general el proceso de software cuenta con cuatro actividades fundamentales las cuales son: Especificación, diseño e implementación, validación y evolución del software.

No obstante esta estructuración del proceso de software los fracasos en los proyectos son evidenciados en todas las latitudes [1] esta ha sido la enfermedad del software desde sus inicios siendo denominada como la “crisis del software” [2] desde los años sesenta. A pesar de la continua búsqueda de soluciones a dicha crisis hoy en día a más de cuarenta años de su primera mención se siguen presentando incontables fracasos, en su mayoría entregas fuera del tiempo estipulado, o fuera del presupuesto e incluso en menor proporción proyectos abandonados los cuales son también evidenciados en el ambiente de desarrollo nacional [3].

Según Sommerville, la especificación del software o ingeniería de requerimientos “consiste en el proceso de comprender y definir qué servicios se requieren del sistema, así como la identificación de las restricciones sobre la operación y diseño del sistema” [4]. Por lo tanto en palabras más coloquiales, durante esta etapa el desarrollador indaga sobre los deseos y necesidades del cliente.

Es en esta búsqueda exhaustiva de la ingeniería de software donde se evidencia que muchos de los errores que se presentan en el desarrollo de un proyecto de software tienen sus orígenes en la obtención de requerimientos y que el valor de corregir estos errores, crece entre más tarde se detecten [5]. Lo que supone que los errores cometidos durante la etapa de adquisición de requerimientos pueden ser los más costosos de corregir.

Para lograr un efectivo conocimiento del negocio y una plena adquisición de los requerimientos de software se deben vencer contados obstáculos, los cuales están relacionados tanto a la efectiva adquisición del conocimiento del dominio como a la interacción con los interesados [6]. Estos obstáculos hasta el momento no logran ser vencidos con las técnicas tradicionales de adquisición de requerimientos, manteniendo a esta actividad como una etapa estancada en cuanto a avances en técnicas innovadoras y que mejoren su efectividad [7].

Esta inminente carencia de información proveniente del cliente mismo o de falencias en la obtención de requisitos por parte del ingeniero de software tiene como consecuencia un “mal entendimiento de los requerimientos”, lo que se define de manera general como falta de conocimiento del negocio, poca comunicación

del conocimiento del dominio por parte de los interesados, una deficiente colaboración con los desarrolladores, ausencia de planificación de las tareas de trabajo con los requerimientos por parte de los desarrolladores, falta de conocimiento de la forma de construir los modelos de requerimientos y ninguna comprensión de que el desarrollo siempre debe ser conducido por los requerimientos. De lo anterior se evidencia que buena parte de los problemas en los proyectos de software se derivan de que el desarrollador inicia un proyecto de software sin conocer ampliamente el dominio de la aplicación y peor aún sin entender completamente las necesidades y expectativas del cliente afectando seriamente la calidad del software [8].

La problemática anterior se presenta especialmente porque el trabajo con los requerimientos se ha desvirtuado a tal punto de verse como una actividad de poca importancia, a la cual se le destinan menos recursos, menos tiempo. Los analistas son vistos incluso como desarrolladores de segunda clase y peor aún en algunos grupos ni siquiera existen. Estos grupos trabajan esperando que la simple presencia eventual del cliente genere por arte de magia listados o diagramas de casos de uso con los requerimientos pertinentes [8]. Debido a esto el cliente considera que al dar un título o nombre a la aplicación ya está todo claro y lo próximo que verá será el resultado final tal cual lo ha imaginado, a su vez el cliente no le da importancia a las jornadas de entrevistas, cuestionarios, revisión de informes, etc. Lo que se puede sintetizar en que el cliente y el diseñador no cuentan con las habilidades, la intención y el tiempo suficientes para aplicar técnicas tradicionales de adquisición de requerimientos.

Este proyecto propone el diseño de una técnica que esencialmente busca maximizar la efectividad del tiempo en la que el desarrollador interactúa con el cliente y mejorar el conocimiento que puede adquirir sobre los requerimientos de una aplicación sin haber interactuado de manera tradicional con los interesados, con el fin de obtener los requerimientos de una aplicación de una manera más eficiente. Esto aprovechando la existencia de aplicaciones similares que cumplen parcialmente las necesidades del cliente y que permiten mediante su análisis conocer y determinar aspectos importantes del dominio y de los requerimientos en sí.

Durante el análisis de las problemáticas anteriormente planteadas se generan inquietudes sobre cuál es la manera de hacer esta mejora. Se nota entonces el potencial que tiene la ingeniería inversa y las aplicaciones heredadas para conocer más sobre los requerimientos de los clientes en determinada aplicación de software que cuente con versiones anteriores o símiles en algunas de sus funcionalidades [9]. Se busca obtener elementos que le den al desarrollador herramientas que le permitan tomar la delantera en la visión que puede tener los clientes de un sistema, así como también poder generar información válida y útil para su trabajo sin tener que depender meramente de disponibilidad de terceros [10]. Cabe aclarar que un software no puede construirse sin la participación de los

interesados, esto es de naturaleza innegable, aunque difícil de medir con exactitud, en general existen efectos positivos confirmados en cuanto a la relación directa entre clientes involucrados y el éxito de los proyectos de software [11], pero si se debe resaltar que poder tener conocimiento de antemano sobre las necesidades del cliente, da al ingeniero de software herramientas poderosas para aprovechar los espacios y obtener la mejor información del cliente al momento de hacer una recolección de requerimientos tradicional.

Es por esto el diseño de una técnica que involucre ingeniería inversa de la interfaz gráfica de aplicaciones heredadas para la obtención de requerimientos es la motivación primaria de esta investigación. Además de encontrar el grado de exploración que existe en este campo y ahondar en el mismo. Abriendo la posibilidad de investigaciones futuras que se sustenten en los resultados obtenidos en este trabajo.

Otra iniciativa que nace simultáneamente con lo planteado anteriormente es que a través de este trabajo se puedan generar nuevas inquietudes sobre la forma en que se realiza la adquisición de requerimientos y la variedad de aplicaciones que presenta la ingeniería inversa. Dejando como producto una técnica usable y un punto de referencia para desarrolladores e investigadores interesados en el área. Esto con el fin de aportar al proceso de software y al mejoramiento del mismo, buscando siempre dar calidad y valor al software.

2. JUSTIFICACIÓN

La Ingeniería de Software busca continuamente nuevas alternativas para mejorar el proceso de software, el estudio continúo de nuevas metodologías, técnicas y herramientas hace parte de las tareas de los investigadores en esta área. Sin embargo las claras problemáticas que se encuentran en el desarrollo de software no son una novedad, los proyectos con entregas tardías, presupuestos insuficientes y peor aun totalmente fallidos, son los principales enemigos del éxito de la aplicación de la ingeniería a la creación de software [12].

Sin embargo, la ingeniería de requerimientos y más específicamente la adquisición de requerimientos es una de las etapas dentro del ciclo del software que menos avances o cambios significativos ha tenido en la forma como se desarrolla (problemas universales). La adquisición de requerimientos en muchos casos se ha visto relegada a un segundo plano, siendo una de las actividades con menos recursos en el proceso de software, generando incontables deficiencias en la implementación y calidad del software [8].

Debido a las deficiencias detectadas en la actividad de recolección de requerimientos durante el proceso de software y a las reducidas alternativas existentes en este campo, nace la necesidad de proponer una técnica alternativa que fortalezca esta actividad y que a su vez proponga el uso de métodos alejados sustancialmente de los que se usan tradicionalmente en adquisición de requerimientos. Los cuales se basan en una fuerte interacción con el usuario.

De esta forma se da al analista de requerimientos de software una herramienta que le permita encontrar requerimientos de software que minimiza la participación del usuario, por medio de la ingeniería inversa de interfaces gráficas de sistemas heredados. Esto con el fin de aumentar la efectividad del proceso tradicional de adquisición de requerimientos y de la implementación en sí. Ya que permite al desarrollador tener un conocimiento previo en cuanto a requerimientos y dominio de la aplicación, aprovechando más el tiempo y disponibilidad de los interesados al efectuar actividades más habituales de obtención de requerimientos.

Esta nueva posibilidad a la hora de adquirir requerimientos de software busca aportar una mejora en el éxito en los proyectos de software, ya que los requerimientos son la base de cualquier proyecto de software, es a través de ellos que se definen el diseño, la tecnología y por supuesto el impacto de la aplicación.
Este trabajo pretende abrir la posibilidad de nuevos estudios sobre la creación de aplicaciones de software con una menor participación de los usuarios, aprovechando herramientas de ingeniería que han estado disponibles por décadas como es el caso de la ingeniería inversa. Para esto el proyecto planteado involucra

a su vez una amplia investigación sobre esta temática y las alternativas existentes lo que complementa el resultado esperado con la síntesis de una literatura disponible al respecto, pudiendo ser una base teórica efectiva para investigadores y profesionales interesados en la posibilidad de conocer los efectos que pueda tener una manera diferente de direccionar la interacción con los clientes en los proyectos de software. Siendo esta investigación de especial interés para el director de este proyecto quien busca ahondar en esta área y las posibilidades que brinda.

Finalmente y en especial como motivación personal, derivada de las vivencias de la autora en su trabajo como desarrolladora de software, se busca obtener elementos que reivindiquen y mejoren la condición actual del desarrollador de software en la industria.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Proponer una técnica para adquirir requerimientos de software mediante el uso de ingeniería inversa aplicada a interfaces gráficas de usuario de sistemas heredados.

3.2 OBJETIVOS ESPECÍFICOS

- Desarrollar un marco conceptual amplio para analizar las posibilidades existentes en el uso de la ingeniería inversa como herramienta para adquirir requerimientos de software, teniendo en cuenta los métodos, técnicas y herramientas halladas en los estudios más sobresalientes y actuales relacionados con dicho fin.
- Identificar las técnicas más utilizadas para la captura de requerimientos de software en el desarrollo de aplicaciones a nivel local, para estudiar en detalle la manera como los desarrolladores de software están manejando la adquisición de requerimientos de software y lo especialmente relacionado con el tiempo de disponibilidad de los clientes para la realización de estas actividades.
- Diseñar una técnica de ingeniería inversa para el uso de software heredado en la adquisición de requerimientos que se ajuste a la situación de la industria de desarrollo de software local.
- Validar la técnica en proyectos que han usado en su desarrollo un proceso de ingeniería directa, para determinar la efectividad de la técnica propuesta en la adquisición de requerimientos de software.

4. MARCO DE REFERENCIA

Uno de los objetivos específicos más importantes planteado en los antecedentes de esta investigación, es el de construir un marco conceptual amplio, el cual tiene como finalidad conocer el estado del arte y reunir los conceptos fundamentales para el desarrollo de este proyecto.

La base teórica para el marco de referencia se fundamenta en los tres estadios descritos de la investigación exploratoria: el estudio de la literatura, la revisión de la experiencia y finalmente el registro organizado de la información. Referente al estudio de la literatura se presentan dos resultados importantes. El primero es la conceptualización de las temáticas necesarias para establecer la técnica propuesta en este proyecto, y el segundo la construcción de un estado del arte estructurado en sus primeras etapas como una ontología. Para la revisión de la experiencia se plantea una búsqueda de campo, la cual tiene como objetivo principal determinar las técnicas de captura de requerimientos utilizadas actualmente en el espacio de la industria de desarrollo de software en el área local. El registro organizado de la información se evidencia en las siguientes secciones de este marco de referencia.

4.1 MARCO TEÓRICO

A continuación se presenta un resumen de los conceptos teóricos que soportan el presente trabajo de investigación, estos temas son todos aquellos conocimientos que han tenido que profundizarse para el desarrollo de la técnica propuesta.

4.1.1 Requerimientos de Software

Los requerimientos de software han sido definidos desde innumerables puntos de vista, una definición bastante simplificada pero que expresa claramente la idea de un requerimiento es la dada por los autores de Head First Object Oriented Analysis and Design los cuales lo definen como “una cosa específica que nuestro sistema debe hacer para funcionar correctamente” [13]. En cuanto a una cosa específica un requerimiento debe entonces verse como una cosa individual, básica, una propiedad que debe tener el software, mientras que, funcionar correctamente está asociado directamente con la capacidad que debe tener el software para resolver el problema del mundo real para el cual fue diseñado.

Un requerimiento tiene como propiedad esencial el ser verificable, esto significa que debe poder comprobarse su correcta implementación dentro del sistema. En algunos casos esta verificación es un proceso costoso ya que puede requerir de la intervención de expertos o la implementación de software de simulación [14].

La definición de requerimientos a su vez se ha especificado como, requerimientos de usuario y requerimientos del sistema. Los requerimientos de usuario son estamentos en un lenguaje natural acompañados de diagramas que describan los servicios que se espera que el sistema entregue a los usuarios y las constantes bajo la cuales debe operar el sistema. En cuanto a los requerimientos del sistema algunas veces también llamados especificaciones funcionales o requisitos, son aquellos que describen funciones, servicios, y constantes operacionales del sistema. La especificación funcional define lo que debe ser implementado y se recomienda haga parte de contrato de desarrollo [4].

4.1.1.1 Clasificación

De igual manera que la definición de requerimientos ha tenido innumerables puntos de vista, su clasificación también ha recorrido un largo camino, desde ninguna clasificación, hasta algunas muy detalladas y extensas. Sin embargo una de las clasificaciones más aceptada y en este caso la que más se ajusta a las consideraciones hechas en esta investigación es la que divide los requerimientos en funcionales y no funcionales.

Requerimientos Funcionales Los requerimientos no funcionales describen claramente las funciones que el software debe ejecutar, esto significa que por ejemplo puntualizan como el sistema reacciona a entradas particulares y como el sistema debe comportarse en cada situación. Sencillamente los requerimientos funcionales definen lo que el sistema debe o no debe hacer [4], [14].

Requerimientos No Funcionales Los requerimientos no funcionales esencialmente son las restricciones sobre el sistema, actúan para restringir todos los aspectos de la solución. Estos requerimientos pueden clasificar a su vez en requerimientos de desempeño, requisitos de mantenimiento, requisitos de seguridad, etc. Estos requerimientos toman al sistema como un todo en lugar de características o servicios individuales [4], [14].

4.1.1.2 Captura de Requerimientos

Es la primera etapa del proceso de requerimientos y existen múltiples definiciones de la misma, en el Swebook se define como todo lo que concierne a de dónde provienen los requerimientos de software y como el ingeniero puede recolectarlos [14]. Sommerville también afirma que “El descubrimiento de requerimientos es el proceso de recopilar información sobre el sistema requerido y los sistemas existentes, así como de separar, a partir de esta información los requerimientos de usuario y del sistema” [4]. Se comprende entonces que la adquisición de requerimientos es una etapa de captura de información para la comprensión del sistema y para entender las necesidades del cliente.

La captura de requerimientos es una actividad asociada fundamentalmente al análisis de requerimientos. Durante estas etapas los ingenieros deben trabajar de manera cercana con los clientes y los usuarios finales, con el fin de aprender sobre el dominio de la aplicación, reconocer los servicios que debe proveer el sistema y determinar cosas como el desempeño del sistema, requerimientos de hardware etc. [4]

4.1.1.3 Fuentes de Requerimientos

Los requerimientos de software se obtienen a través de múltiples actividades, es esencial identificar todas las fuentes potenciales y detectar el impacto que puedan tener sobre el proyecto de software. Entre las fuentes más importantes y generalmente identificadas por los ingenieros de requerimientos, se encuentran:

- Las metas: también llamados asuntos de interés del negocio, y se refieren a los objetivos generales y de alto nivel del software. Dan el motivo al software. El ingeniero de requerimientos debe prestar especial atención a la prioridad y el valor dado a estas metas, ya que generalmente están vagamente formuladas.
- Conocimiento del dominio: el ingeniero de requerimientos debe adquirir o tener conocimiento sobre el dominio de la aplicación, lo que le permite inferir conocimiento tácito que los interesados muchas veces no logran articular.
- Interesados: el ingeniero de requerimientos necesita identificar, representar y administrar los puntos de vista de muchos tipos de interesados.
- Ambiente operacional: los requerimientos se derivan del ambiente en el cual el software será ejecutado. Debe ser cuidadosamente identificado ya que puede afectar la factibilidad, el costo y las restricciones de diseño del software.
- Ambiente organizacional: el software es generalmente requerido para apoyar un proceso de negocio, el ingeniero de requerimientos debe ser muy cuidadoso teniendo en cuenta la estructura, la cultura, y políticas organizacionales.
- Sistemas existentes que hayan sido actualizados.
- Sistemas análogos.

4.1.1.4 Técnicas de Captura de Requerimientos

A continuación se muestran las principales técnicas de exploración utilizadas hoy por los ingenieros de requerimientos para capturar los requerimientos de las fuentes mencionadas anteriormente. Entre ellas se encuentran el muestreo de la documentación, las formas y las bases de datos existentes, investigación y visitas al sitio, observación del ambiente de trabajo, cuestionarios, entrevistas, propuestas de prototipos y planeación conjunta de requerimientos.

En la mayoría de proyectos de software se aplican varias técnicas, la selección de estas técnicas varía de acuerdo al tipo de aplicación y al conocimiento del ingeniero de requerimientos sobre las ventajas y desventajas de cada una de las técnicas existentes.

Muestreo de la documentación existente, formatos y archivos existentes Si ya hay un sistema existente el analista de sistemas puede utilizar la consulta de la documentación, formatos y archivos históricos. Siempre se debe considerar primero la documentación existente antes que las personas. Dentro de la documentación existente que debe tener en cuenta el ingeniero de requerimientos se encuentra el organigrama, historial que originó el proyecto también se revisan diagramas de flujo, diccionarios o repositorios, documentación de diseño, documentación de programa, manuales de operación y manuales de entrenamiento. Para analizar toda esta información el ingeniero de requerimientos debe utilizar técnicas de muestreo, ya que sería impráctico estudiar todas las ocurrencias de todos los formatos o registros [15].

Investigación y visitas al sitio Otra técnica de exploración se basa en realizar investigación sobre la problemática que se presenta y cómo esta ha tratado de ser solucionada por otros. Revistas especializadas, libros de referencia etc. Son una fuente segura y viable de información. También visitar otras compañías o departamentos que han encarado problemas similares [15].

Algunas de las ventajas de la observación son que los datos recolectados pueden ser muy confiables y se usan para verificar la validez de los datos obtenidos directamente de las personas, a su vez el analista observa exactamente lo que se está haciendo, identifica tareas omitidas, obtiene datos que describen el ambiente operacional físico, es una técnica de un costo bajo y mediante la misma el analista puede realizar mediciones de la actividad observada.

Entre las desventajas que tiene la técnica de observación se encuentran, que los individuos observados se pueden sentir incómodos y no tener un comportamiento natural, también el trabajo que se está observando puede tener una variación en la dificultad que tiene durante el tiempo que se observa, además muchas de las tareas pueden realizarse en horarios no convencionales lo que se convierte en una incomodidad para el analista, las tareas pueden estar sujetas a interrupciones, y ser realizadas de una forma correcta o incorrecta.

Cuestionarios Los cuestionarios son una forma de encuestas conducidas mediante cuestionarios, estos se pueden producir de manera masiva, recolectando hechos de un gran número de personas al mismo tiempo. Este documento permite al ingeniero recolectar información y opiniones de los encuestados. Han sido criticados por el uso inapropiado que se ha hecho de ello y

se recomienda al ingeniero conocer muy bien sus ventajas y desventajas antes de su uso. A continuación se mencionan algunas de sus ventajas y desventajas:

Tabla 1. Ventajas y desventajas de los cuestionarios

Ventajas	Desventajas
La Mayoría de los cuestionarios pueden responderse rápidamente. La gente puede completar y devolver los cuestionarios con toda comodidad.	Con frecuencia el número de encuestados es bajo.
Los cuestionarios son un medio económico de recopilar datos provenientes de un gran número de personas.	Los cuestionarios son inflexibles, no hay oportunidad que se obtenga información voluntaria.
Los cuestionarios permiten que las personas mantengan el anonimato. Hay más probabilidad que suministren hechos reales.	No es posible observar y analizar el lenguaje corporal del encuestado.
Las respuestas pueden tabularse y analizarse rápidamente.	No hay una oportunidad inmediata para aclarar una respuesta vaga o incompleta a cualquier pregunta.
	Los buenos cuestionarios son difíciles de preparar.

Fuente: J. Whitten L. and L. Bentley D., Análisis de sistemas: diseño y métodos [15]

Entrevistas Las entrevistas son la técnica más importante y la que más se utiliza al momento de obtener información sobre los requerimientos de un sistema, generalmente el ingeniero recolecta información mediante la interacción directa con las personas interesadas. El esquema básico de esta técnica consiste en que el usuario o propietario del sistema es el entrevistado y el ingeniero de requerimientos es el entrevistador, al entrevistado se le solicita que responda una serie de preguntas que proporcionen información relevante para el levantamiento de requerimientos y pueden haber tanto múltiples entrevistados como múltiples entrevistadores. Generalmente los objetivos que busca la técnica de la entrevista son indagar, verificar y aclarar hechos, generar entusiasmo, involucrar al usuario final, identificar los requerimientos y preguntar sobre ideas y opiniones.

Tabla 2. Ventajas y desventajas de las entrevistas

Ventajas	Desventajas
Motivan al entrevistado para que responda libre y abiertamente. Le da al entrevistado una percepción de que está contribuyendo activamente al proyecto de sistemas.	Consume mucho tiempo y por lo tanto es muy costosa.
Permiten obtener más retroalimentación del entrevistado.	El éxito de las entrevistas depende de las habilidades en relaciones humanas del analista.
Permiten adaptar o parafrasee las preguntas para cada persona.	Pueden ser imprácticas debido a la ubicación del entrevistado.
Dan al analista la oportunidad de observar la comunicación no oral del entrevistado.	

Fuente: J. Whitten L. and L. Bentley D., Análisis de sistemas: diseño y métodos [15]

Elaboración de prototipos de identificación La elaboración de prototipos de identificación es una técnica de exploración que consiste en la elaboración de un pequeño modelo de trabajo de los requerimientos de usuario o del diseño propuesto para el sistema de información. Entre sus pros y contras se encuentran los descritos en la tabla 3:

Tabla 3. Ventajas y desventajas de los prototipos de identificación

Ventajas	Desventajas
Permite que los usuarios y desarrolladores experimenten con el software y desarrollen una comprensión de cómo podría trabajar el sistema.	Puede ser necesario entrenar a los desarrolladores en el enfoque de elaboración de prototipos.
Ayuda a determinar la factibilidad y la utilidad del sistema antes de incurrir en altos costos de desarrollo.	Los usuarios pueden desarrollar expectativas poco realistas basándose en el desempeño, la confiabilidad y las características del prototipo.
Sirve como un mecanismo de entrenamiento de los usuarios.	La elaboración de prototipos puede prolongar el programa de desarrollo y aumentar los costos.
Ayuda a construir los planes y escenarios de prueba del sistema.	
Puede minimizar el tiempo invertido en la exploración y ayuda a definir requerimientos más estables y confiables.	

Fuente: J. Whitten L. and L. Bentley D., Análisis de sistemas: diseño y métodos [15]

Talleres de requerimientos Los talleres de requerimientos son una forma alternativa y rápida de capturar requerimientos. En ella los interesados participan dentro de un ambiente propicio dándose cuenta que el trabajo de capturar requerimientos no tiene por que ser una tarea extenuante o que tome demasiado tiempo. Un taller de requerimientos debe contar con una estructura iterativa. La posibilidad de interacción entre diferentes tipos de interesados añade un valor agregado a los requerimientos, ya que un taller como este puede ser la primera vez que un grupo de interesados este reunido y pueda interactuar, conduciendo la creación de requerimientos que aporten una mejor percepción de lo que cada grupo desea el sistema esté en capacidad de hacer y cómo estas características afectan las necesidades de los demás grupos [16].

4.1.2 Interfaz Gráfica de Usuario

Para poder comprender mejor el concepto de interfaz gráfica de usuario es recomendable comprender algunos otros conceptos previamente.

Interfaz de Usuario La interfaz de usuario es el mecanismo mediante el cual el usuario se comunica con el computador, es decir provee tanto el mecanismo de entrada, donde el usuario le “dice” al computador qué hacer, como el mecanismo de salida, a través del cual el computador responde a usuario. Las personas interactúan con un computador a través de interfaz usando el teclado, ratón, pantallas táctiles y micrófonos [17].

Interfaz de Línea de Comandos (CLI Command Line Interface) Es una interfaz de usuario no gráfica donde el usuario debe ingresar comandos para interactuar con la aplicación. Se considera entonces una interfaz basada en texto orientada al teclado donde el usuario digita una línea de comandos con parámetros, entonces presiona entrar para ejecutar. La interfaz puede ser interactiva, en este caso el usuario es incitado a ingresar más comandos en una secuencia, o no interactiva, donde el programa se ejecuta sin ninguna interacción de usuario. La interfaz de línea de comandos es popular en procesos por lotes, cuando una operación simple debe aplicarse múltiples veces [17].

Interfaz WIMP (WIMP Windows – Icons – and Pointer- Based Interface) Una interfaz WIMP es una temprana evolución de la GUI, la cual se basa en el uso de un ratón, junto con elementos clave de interfaz de usuario, ventanas, iconos, y menús desplegables. Las ventanas contienen los programas que se están ejecutando, y se entiende que debe darse clic a los iconos iniciar la ejecución de dichos programas, los menús proveen una lista de comandos disponibles, y los punteros (cursores) permiten al usuario rastrear visualmente el ratón [17].

Interfaz Gráfica de Usuario Una interfaz gráfica de usuario, o GUI, es la evolución de la interfaz que incluye elementos reusables de la UI y es soportada por dispositivos móviles, la cual no necesariamente usa puntero de ratón. Esta permite interactuar con la aplicación usando gráficos, imágenes, iconos, y elementos en la pantalla 2-D, sin tener que memorizar complejos comandos y digitarlos de manera precisa usando el teclado, tal como en la CLI [17].

4.1.3 Diseño de Interfaces Gráficas

El diseño de interfaces gráficas de usuario no es una tarea meramente visual, donde se determine mecánicamente los colores, tipos de letra y formas de los componentes de interfaz. El diseño de interfaces gráficas de usuario inicia "entendiendo a la gente", cuando se piensa en cómo debe percibirse la interfaz de usuario de un sistema se debe primero pensar en lo que le gusta a la gente, ¿por qué motivos usan una determinada herramienta de software?, y ¿cómo van a interactuar con dicha herramienta? El software que soporta el comportamiento humano ayuda de una mejor forma al usuario a lograr sus metas, ya que un usuario únicamente usa una herramienta sea software o no cuando tiene una razón específica para usarla. Es por esto que sin importar lo llamativa, dinámica o colorida que sea una interfaz gráfica, esta es exitosa únicamente si logra resolver de manera sencilla y ágil "el problema correcto" para el usuario [18].

Entonces es importante como primer paso en el diseño de una GUI, preguntarse ¿qué pretenden lograr los usuarios con la herramienta software en construcción? Determinar cuál es el tipo de transacción que desean realizar. Las razones más comunes que tienen los usuarios son por ejemplo:

- Encontrar algún dato u objeto.
- Aprender algo
- Realizar una transacción
- Controlar o monitorear algo
- Crear algo
- Conversar con otras personas
- Entretenerse

Para identificar estas metas y conectarlas de manera lógica con el diseño se debe realizar una buena cantidad de “investigación del usuario”, la cual permite obtener información relevante sobre el usuario, caracterizando el tipo de personas que usaran dicho diseño. Sin embargo, es claro que no todas las personas son iguales, pero, que es imposible personalizar un diseño acorde a las características únicas de cada individuo. El éxito se logra descubriendo generalidades sobre los usuarios, es decir aprender lo suficiente sobre los usuarios individualmente para lograr separar las peculiaridades de los patrones comunes de comportamiento [18].

Específicamente lo que se quiere aprender sobre los usuarios y que ayuda a determinar las generalidades que los caracterizan es:

- Sus metas al usar el software o sitio web
- Las tareas específicas que emprenden en busca de dichas metas
- El lenguaje y las palabras que usan para describir lo que están haciendo
- Su habilidad al usar un software similar al que se está diseñando
- Sus actitudes hacia el diseño y como los diferentes diseños afectan esas actitudes

Esta fase es tediosa, consume tiempo y recursos pero es necesaria para lograr diseños bien aceptados y eficaces [18].

4.1.3.1 Arquitectura de la Información (AI)

Después de aprender sobre lo que los usuarios quieren, es necesario pensar en la aplicación en términos de los datos subyacentes y las tareas. Cosas tales como la forma en que se van a mostrar los objetos, cómo están categorizados y organizados, qué cosa van a hacer los usuarios con ellos y cómo estos objetos pueden ser presentados de muchas formas en el diseño, son los elementos que determinan la organización del contenido. Esto hace referencia específicamente a cómo se organiza el contenido de la aplicación.

En este punto entra en escena lo que se denomina Arquitectura de la Información (AI), la cual es el arte de organizar un espacio de información, lo cual abarca la presentación, búsqueda, navegación, etiquetado, categorización, organización, manipulación y encubrimiento estratégico de la información [18].

4.1.3.2 Modelo de Interacción

Una aplicación o sitio web generalmente cuenta con un servicio o múltiples servicios tales como compartir información, vender un producto o marca, comunicación o un incontable número de metas. Una aplicación tiene una o

múltiples páginas para realizar las tareas que le permitan cumplir con dichos servicios. Cualquier página que haga parte de una aplicación realizará primariamente una de las siguientes cosas:

- Mostrar una sola cosa, como un mapa, libro, video o juego
- Mostrar una lista o conjunto de cosas
- Proveer herramientas para crear una cosa
- Facilitar una tarea

Lo anterior se consideran modelos de interacción y establecen consistencia a través del artefacto, y determina como los usuarios se mueven a través y entre las diferentes piezas de funcionalidad. La mayoría de las páginas usan por lo menos uno de estos o se conforman más comúnmente de una combinación de estos modelos [18].

Cuando la naturaleza de la aplicación parece tomar la dirección de alguno de estos modelos, se recurre a los patrones de diseño que corresponden a estos modelos. En el diseño de aplicaciones y gracias a la observación y múltiples estudios se logran identificar tendencias entre la aplicación y los usuarios, estas han producido múltiples patrones tanto de interacción, como relacionados con la forma de recolección de datos y la forma en que se desea mostrar la información.

En este caso se presentan a continuación los modelos de interacción mencionados anteriormente y los patrones de diseño que pueden ser usados a la hora de implementar una aplicación o sitio web.

Mostrar una sola cosa Todo el diseño de la página muestra o despliega una pieza individual de contenido, sin ninguna otra cosa que los usuarios puedan ver. Todo lo que se necesita es manejar la interacción que tiene el usuario con esa única cosa. La AI es probablemente directa. En este tipo de diseño puede haber herramientas a pequeña escala alrededor del contenido, *scrollers* y deslizadores, caja de registro, navegación global, encabezados y pies de página etc. Sin embargo estos deben ser diseñados de manera muy simple.

Mostrar una lista de cosas Esta es una de las cosas que la mayoría de aplicaciones parece realizar. El mundo digital ha convergido con muchas formas para mostrar listas, la mayoría de las cuales son comunes para casi todos los usuarios. Listas de texto simple, menús, grillas de imágenes, resultados de búsquedas, listas de correos electrónicos u otras comunicaciones, tablas, árboles, etc. El uso de listas presenta retos en la AI, asociados a su longitud, jerarquía, ordenamiento, filtros de búsqueda y las operaciones asociadas a cada uno de los elementos.

Proveer herramientas para crear una cosa Constructores y editores son el dominio predominante en las clases de software en el mundo. Herramientas ofimáticas, de diseño, editores de código, ambientes de desarrollo integrado, editores de imagen y hojas de cálculo, son ampliamente usados y con estilos de interacción bien establecidos.

Facilitar una tarea En algunos casos la interfaz no debe mostrar grandes a las listas o servir para construir cosas complejas, en algunos casos simplemente deben realizarse tareas simples, registrarse, postear, imprimir, cargar información, comprar, cambiar una configuración. No es necesaria demasiada AI si el usuario puede realizar el trabajo necesario en una pequeña área, sin embargo si la tarea se hace más complicada es necesario estructurarla.

En la tabla 4 se presentan los patrones de diseño de GUI asociados a cada uno de los modelos de interacción mencionados anteriormente.

Tabla 4. Modelos de interacción y patrones de diseño

Modelo de interacción	**Patrón de diseño de GUI**							
	Múltiples espacio de trabajo	Destacado, búsqueda y navegación	Corriente de Noticias	Administrador de imágenes	Vistas alternativas	Lienzo y paleta	Asistente	Editor de configuración
Mostrar una sola cosa	x				x			
Mostrar una lista de cosas		x	x	x				
Proveer herramientas para construir una cosa	x				x	x		
Facilitar una tarea							x	x

Estos patrones permiten tener una idea del diseño de una interfaz gráfica de acuerdo a los modelos de interacción, las características fundamentales de los patrones se muestran en la tabla 5.

Tabla 5. Características de patrones de diseño GUI

Patrones de Interfaces Gráficas de usuario	
Patrón	**Características**
Múltiples Espacios de Trabajo	La aplicación permite desarrollar múltiples tareas al tiempo, el usuario puede ir de un proceso a otro sin tener que finalizar alguno de los procesos, este patrón se distingue por la presencia de pestañas (tabs) en el nivel superior, grupos de pestañas, múltiples ventanas donde los usuarios pueden ver diferentes documentos, proyectos, archivos o contextos al mismo tiempo. (Pestañas, ventanas de sistema operativo separadas, columnas o paneles dentro de una ventana, ventanas divididas, etc.)
Búsqueda destacado y navegación	El sitio o aplicación contiene tres elementos obligatoriamente, un elemento destacado, una caja de búsqueda y una lista de elementos o categorías navegables. La aplicación posee una lista extensa de elementos que interesan al usuario, debe facilitar su búsqueda, a su vez alguno de estos elementos debe ser destacado ya que puede ser de interés para el usuario. (Sitios o agencias de noticias).
Corriente de Noticias	Muestra elementos organizados cronológicamente, esta característica denota que los elementos son afectados por el tiempo, los últimos y considerados de mayor importancia se ubican en la parte superior, actualiza dinámicamente y puede combinar elementos de múltiples fuentes.
Administrador de Imágenes	Este tipo de patrón muestra tres tipos de patrones más simples trabajando juntos, miniaturas, vista de ítems e interfaz de búsqueda. La grilla de miniaturas muestra una secuencia de elementos (en este caso imágenes), la vista individual de ítem muestra una versión grande de la imagen seleccionada de la grilla de miniaturas, y la interfaz ofrece usualmente una caja de búsqueda además de otras herramientas de filtrado.

Vistas Alternativas	La aplicación permite que el usuario escoja entre vistas alternativas que son substancialmente diferentes de la vista por defecto. La interfaz cuenta con un "interruptor" para cambiar el modo de visualización. En estas vistas alternativas alguna información puede ser añadida y alguna puede ser retirada, sin embargo el contenido básico sigue siendo el mismo.
Lienzo y Paleta	Una paleta de íconos ubicada junto a un lienzo vacío, el usuario da clic sobre los botones de la paleta para crear objetos dentro del lienzo. La paleta por sí misma es una grilla de botones usualmente localizada en la parte superior izquierda del lienzo. Este patrón es usado en cualquier tipo de editor gráfico.
Asistente	Este patrón guía al usuario a través de un conjunto de vistas paso a paso con el fin de realizar una tarea en un orden prescrito. La implementación más obvia y conocida de este patrón presenta cada paso en una página separada, las cuales pueden ser visitadas por medio de los botones Anterior y Siguiente.
Editor de Configuración	Este tipo de patrón es una página o ventana independiente donde los usuarios pude cambiar configuraciones, preferencias, y propiedades. Este tipo de funcionalidad es fácil de encontrar y la mayoría de plataformas tienen un lugar estándar preferencias generales de aplicación, usualmente las esquinas superiores izquierdas o derechas junto al nombre de usuario.

Fuente: J. Tidwell, Designing Interfaces [18].

4.1.3.3 Modelos de Navegación

Los modelos de navegación son la forma en la que las pantallas (páginas o espacios) se unen unas a otras, y como los usuarios se mueven a través de ellas en un sistema de software [18].

A continuación se presentan los modelos de navegación más comunes en sitios web y aplicaciones de escritorio y móviles.

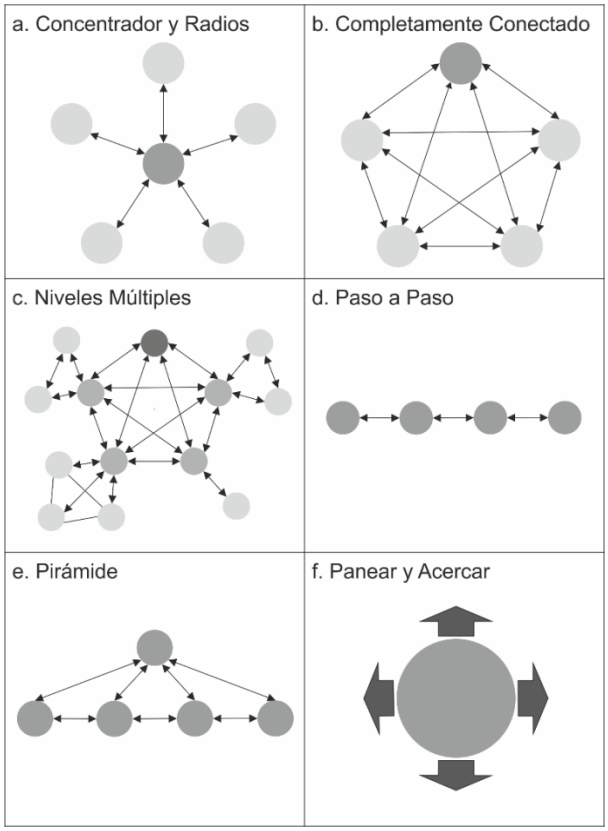
- **Concentrador y Radios (Hub and Spoke)** La mayoría de dispositivos móviles usan esta arquitectura, la cual concentra todas las partes mayores de un sitio o aplicación en la pantalla de inicio. Figura 1a. El usuario puede acceder a cualquiera de estas partes mayores únicamente desde la pantalla principal, esto quiere decir que para moverse a alguna otra pantalla, el usuario siempre debe regresar a la pantalla “central” y de ahí saltar a cualquier otra pantalla.
- **Completamente conectado (Fully Connected)** Muchos sitios web siguen este modelo, en este esquema existe una pantalla o página de inicio como en el modelo anterior, pero todas y cada y de las páginas se conectan entre sí. Las páginas cuentan todas con una característica de navegación global como un menú superior. La navegación global puede ser en un nivel simple como se ve en la figura 1b, con sólo cinco páginas o más profundo con niveles múltiples.
- **Niveles Múltiples *(Multi-level)*** También muy común en los sititos web, las páginas principales están totalmente conectadas, pero las subpáginas están únicamente conectadas entre ellas mismas y usualmente a otras páginas principales, figura 1c.
- **Paso a Paso *(Stepwise)*** Las presentaciones de diapositivas, flujos de proceso y asistentes guían el usuario paso a paso a través de pantallas en una secuencia prescrita. Figura 1d.
- **Pirámide *(Pyramid)*** Es una variante del modelo paso a paso, una pirámide usa una página central o página de menú para listar una secuencia entera

de elementos o subpáginas. El usuario puede seleccionar cualquier ítem dentro de la secuencia y a su vez tiene la opción de usar los enlaces Atrás/Adelante para avanzar a otros ítems en el orden. En cualquier momento puede regresar a la página central. Figura 1e.

- **Panear y Acercar (*Pan-and-Zoom)*** Algunos artefactos son mejor representados como grandes espacios individuales y no como muchos pequeños. Mapas, imágenes grandes, documentos de texto grandes, gráficos de información y representaciones multimedia basadas en tiempo (sonido y video) caen dentro de esta categoría. La navegación panear y acercar ofrece controles para panear (moverse horizontal y verticalmente), acercarse, alejarse y resetear a una posición y estado conocidos. Figura 1f.
- **Navegación Plana *(Flat Navigation)*** Algunas aplicaciones requieren muy poca navegación, un ejemplo puedeser Photoshop[5] u otras aplicaciones complejas que ofrecen gran cantidad de herramientas y funciones que son alcanzables a través de menús, barras de herramientas y paletas [18].

[5] Adobe Photoshop es un editor de gráficos desarrollado por Adobe Systems Incorporated.

Figura 1. Modelos de Navegación



Fuente: J. Tidwell, Designing Interfaces [18].

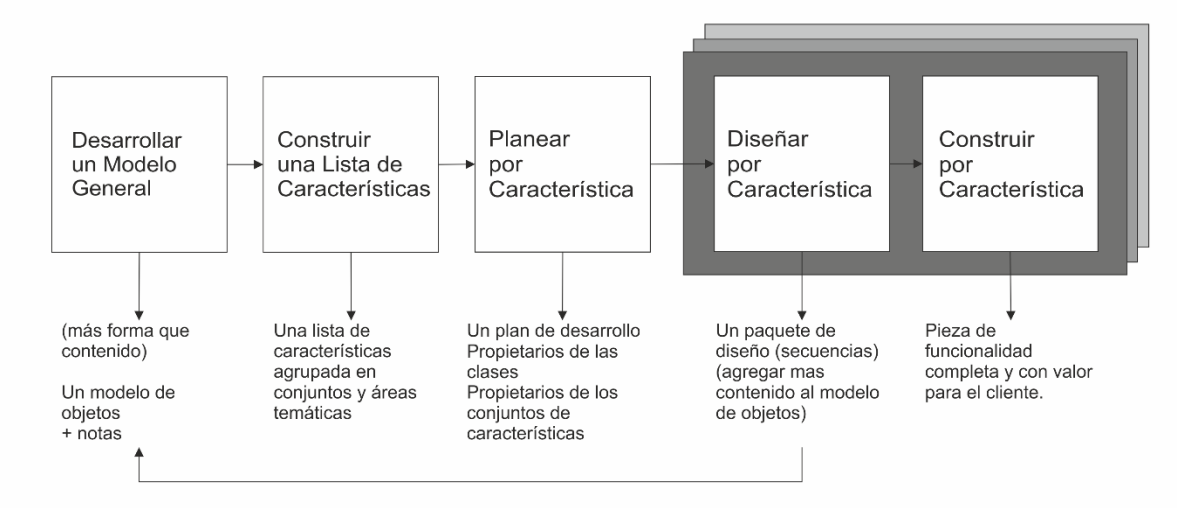
4.1.4 Desarrollo Orientado a Características

El Desarrollo Orientado a las Características (Feature Driven Development, FDD) es una metodología de software ágil y adaptativo que busca diseñar y probar con el fin de entregar resultados de trabajo frecuentemente. El FDD es un proceso altamente iterativo, se enfoca en la calidad en cada paso, entrega resultados de trabajo tangibles y provee información exacta y significativa sobre el estatus del proyecto [19].

FDD inicia con la creación de un modelo del objeto del dominio en colaboración de los Expertos del Dominio. Usando la información recolectada en la actividad de modelado y de otras actividades de requerimientos, los desarrolladores inician la creación de la lista de características. Entonces un somero plan es esbozado y las responsabilidades son asignadas. De esta forma se toman pequeños grupos de características que se encuentren a través del diseño y se construye una iteración, la cual no debe durar más de dos semanas para cada grupo de características, y en la mayoría de los casos es mucho más corta (posiblemente algunas horas). El proceso se repite hasta que no resten más características [19].

La metodología FDD consiste más formalmente de cinco procesos figura 2, este modelo es ajustable de ser necesario durante el desarrollo.

Figura 2. Los cinco procesos de la FDD con sus salidas



Fuente: R. S. Palmer and M. J. Felsing, A Practical Guide to Feature-Driven Development [19]

4.1.4.1 Construcción Lista de Características

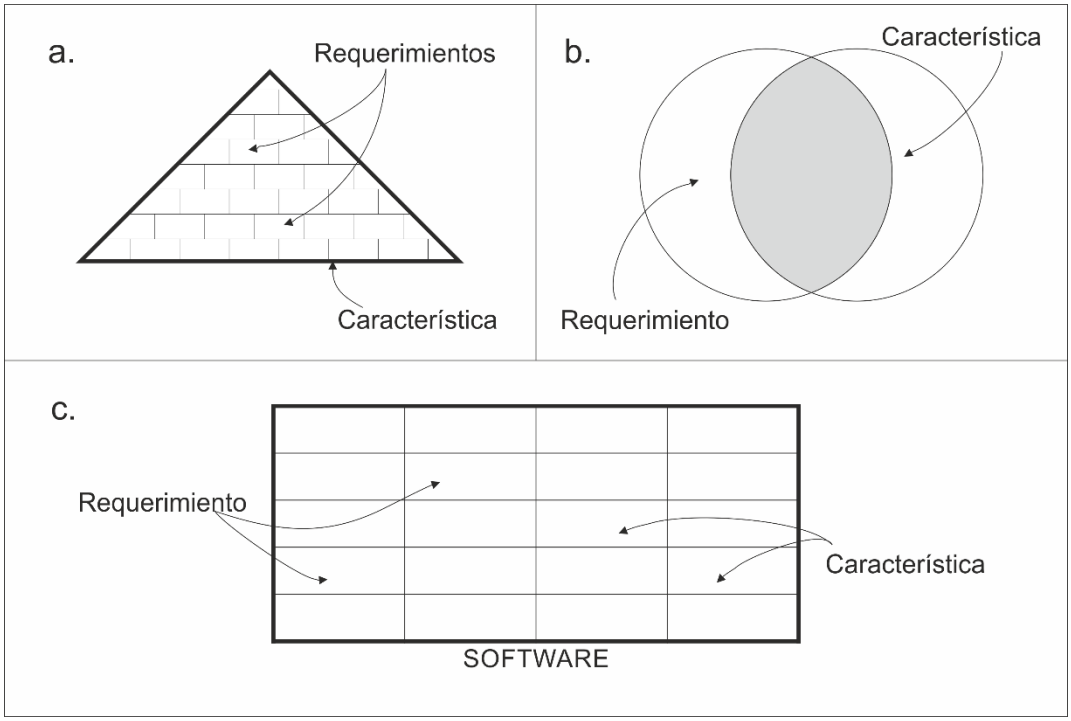
El segundo proceso de la FDD es la construcción de la lista de características, el equipo de modelado está ahora en posición de crear una lista de características para el sistema. La lista está organizada jerárquicamente y será usada como plan, camino y reporte del progreso del equipo de desarrollo a medida que ellos realizan el trabajo iterativo entre los procesos cuatro y cinco de la FDD [19].

La importancia para los fines de esta investigación es reconocer como se construyen y funcionan estas características, y la estructura jerárquica bajo la que se organizan.

¿Qué es una característica? Una característica es una descripción de alto nivel de algo que el sistema necesita hacer. Las características se obtienen generalmente mediante interacciones con los clientes.

Las características y los requerimientos han sido estudiados desde variados enfoques, planteándose diferencias y similitudes entre características y requerimientos. Para algunos autores, una característica es una “gran cosa” que el sistema hace. Para lograr realizar esta “gran cosa”, se requieren múltiples “pequeñas cosas” que el sistema debe hacer. Estas “pequeñas” funciones son consideradas requerimientos. Entonces desde este punto de vista una característica es satisfecha por múltiples requerimientos, ver figura 3a.

Figura 3. Enfoques Características

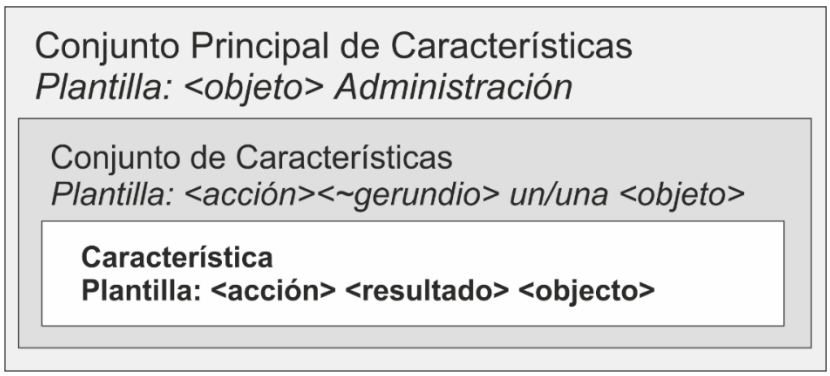


Fuente: G. P. & D. W. Brett D. McLaughlin, Head First Object-Oriented Analysis and Design [13]

Desde otro punto de vista, no se propone una diferencia tan marcada entre característica y requerimiento. Una característica puede ser muy general o muy específica. Para este enfoque los dos conceptos se superponen, lo que supone de cierta forma que los términos pueden llegar a intercambiarse, figura 3b. Sin embargo, no es posible afirmar que uno de los dos enfoques es completamente correcto, en general se recomienda pensar en las características y requerimientos simplemente como cosas que el sistema necesita hacer, figura 3c.

Lista de Características La construcción de la lista de características se realiza con el fin de identificar todos los requerimientos de un sistema. El criterio de salida para este proceso de la FDD es obtener una lista de características estructurada que consiste de una lista de **conjuntos principales de características** o los que pueden identificarse como áreas del sistema, estos conjuntos principales están conformados por lo que se denomina **conjuntos de características** los cuales representan las actividades del negocio. Finalmente esos conjuntos de características se forman de diversas **características**, cada una de estas características individualmente es un paso o actividad menor del conjunto de características. En la figura (), se observa más claramente esta estructura.

Figura 4. Estructura y Plantillas Lista de Características



Fuente: R. S. Palmer and M. J. Felsing, A Practical Guide to Feature-Driven Development [19]

Las características son funciones granulares expresadas en términos valorados por el cliente usando la estructura siguiente: <acción><resultado><objeto>. A su vez existen algunas convenciones para nombrar lo niveles más altos en la jerarquía de la lista de características, para los conjuntos de características se tiene <acción><~gerundio> un/una <objeto> y en el caso de los conjuntos principales de características <objeto> Administración, es decir simplemente el objeto acompañado de la palabra “Administración”. Estas plantillas no son camisa de fuerza y depende de la intuición y pericia de ingeniero el realizar los cambios necesarios para obtener resultados óptimos, en la figura 4 también se pueden ver las plantillas para cada nivel de la lista de características.

4.1.5 Ingeniería Inversa

Una amplia definición de la ingeniería inversa dada en [20], se define como el proceso de extraer el conocimiento o modelo del diseño de cualquier cosa hecha por el hombre. El concepto fue definido mucho antes de lo computadores o la tecnología moderna, y probablemente se remonta a los días de la revolución industrial. Se puede afirmar que la ingeniería inversa tiene similitud con la investigación científica, en esta el investigador trata de resolver modelos de fenómenos naturales, mientras que en la ingeniería inversa el artefacto estudiado es puramente hecho por el hombre. En sus inicios esa investigación de los artefactos construidos por el hombre se ligaba estrechamente a la curiosidad que generaban los primeros artefactos electrónicos y el interés por saber que había dentro de la “caja”, sin embargo actualmente la miniaturización de la tecnología hace poco probable identificar algún elemento claramente simplemente “abriendo la caja”.

El propósito general de la ingeniería inversa es obtener conocimiento que se ha perdido, ideas y decisiones del diseño cuando esta información no está disponible de manera evidente, usualmente porque dicha información está en poder de terceros o simplemente se ha perdido o destruido con el paso del tiempo.

4.1.5.1 Ingeniería Inversa de Software

Hoy en día el software es una de las tecnologías más complejas e intrigantes, la ingeniería inversa de software se encarga de abrir la “caja” de un programa y mirar dentro. Este proceso requiere únicamente de dos herramientas, una computadora y la mente humana [20].

Más específicamente desde el punto de vista del software y los sistemas computacionales, la ingeniería inversa es el proceso de analizar artefactos de software disponibles, tales como requerimientos, diseño, arquitecturas, código y código byte, con el objetivo de extraer información y proveer vistas de alto nivel del sistema en cuestión [21].

La práctica más común en la ingeniería inversa es explotar el código fuente como la descripción más confiable tanto del comportamiento de un sistema de software como de la estructura del negocio y sus reglas. Sin embargo, la ingeniería inversa está inmersa en una variedad de tareas relacionadas a la comprensión y modificación del software tales como la re-documentación de programas y bases de datos relacionales, recuperación de arquitecturas, recuperación de vistas de diseño alternativas, recuperación de patrones de diseño, encontrar la trazabilidad entre el código y el diseño, modernización de interfaces o la extracción del código fuente u otras abstracciones de más alto nivel a partir del código byte cuando el código fuente no está disponible [21].

4.2 REVISIÓN DE LA EXPERCIENCIA

En esta sección se presenta el resultado de la indagación realizada en diez empresas del sector de desarrollo de soluciones de software. El objetivo principal de esta encuesta es reconocer cuales son las Técnicas de Obtención de Requerimientos que cuentan con más popularidad a nivel local.

A continuación se describen cada una de las preguntas realizadas en la encuesta y una descripción de los resultados obtenidos, el formato de encuesta y los resultados específicos se encuentran en el Anexo B de este documento.

La encuesta se enfoca en tres aspectos importantes de interés. Uno, es la cantidad y calidad del tiempo que se dispone para interactuar con los clientes o usuarios del sistema, entonces se busca saber la cantidad de tiempo dedicada a la observación directa y a la realización de entrevistas. Otro aspecto es el uso de otras técnicas populares en la captura de requerimientos como el uso de prototipos, cuestionarios y especialmente análisis de software heredado. El último aspecto es conocer la percepción que se tiene de la efectividad del proceso de captura de requerimientos en estas diez empresas.

En cuanto a la cantidad y calidad del tiempo que se dispone para interactuar con los clientes o usuarios, se tiene que el tiempo de observación directa de tareas del usuario final no sigue una tendencia constante sino que varía entre las diferentes empresas encuestadas. Cabe destacar que en cinco de las diez empresas este tiempo es menor a tres horas semanales o no se aplica. Además la mitad de las empresas manifiestan que el tiempo con los usuarios es suficiente pero esta cantidad disminuye un diez por ciento al referirse a cuál fue la calidad de ese tiempo con el usuario.

Por otra parte y también relacionado con el tiempo con el cliente o usuario se tiene en cuenta las entrevistas. Todas las empresas declaran su uso, sin embargo cabe resaltar que de las diez, tres declaran tener solo entre ocho y veinticuatro horas en total para todo el proyecto, mientras que las demás apenas alcanzan dos horas semanales. Solo dos empresas afirman tener entre dos y ocho horas de entrevistas a la semana. De lo anterior se puede inferir que una de las herramientas más populares y de mayor uso en los proyectos de software para la obtención de requerimientos cuenta con muy poco tiempo para su realización.

En cuanto a lo relacionado con las técnicas de captura de requerimientos se encuentra que la todas las empresas encuestadas utilizan la construcción de prototipos funcionales (evolutivos), y en solo una de ellas además de estos se utilizan también los prototipos de papel y los no funcionales. En cuanto a la finalidad de la construcción de prototipos, todas las empresas concuerdan en el uso de prototipos para validar requerimientos de software, en dos de ellas además se usan para realizar pruebas de concepto pero en ninguna se menciona como una herramienta para obtener requerimientos de software. Respecto a los cuestionarios, estos son poco populares y solo dos empresas mencionan que sean utilizados como herramienta para la obtención de requerimientos.

Para el análisis de software heredado, se evidencia que todas las empresas lo utilizan y que en general se enfocan en el análisis de código y datos especialmente, dejando como último el análisis de la interfaz gráfica de usuario.

Finalmente a lo relacionado con la percepción de la efectividad en el proceso, solo dos de las empresas perciben que el proceso que realizan para la captura de requerimientos está siendo efectivo y que no existen problemas posteriores debido a errores en la captura.

4.3 ESTADO DEL ARTE

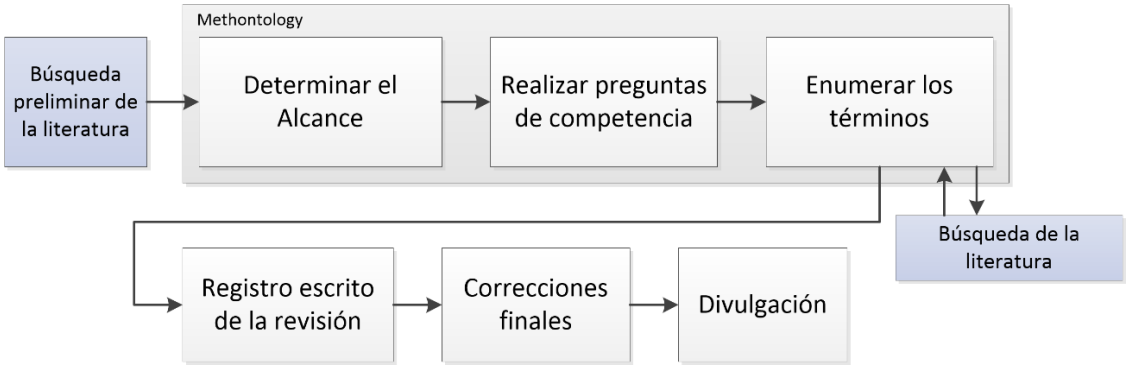
El interés por la investigación en el área de ingeniería inversa de interfaces gráficas de usuario, y la identificación de las alternativas existentes para encontrar elementos de software que puedan ser reutilizados conceptualmente en el emprendimiento de nuevos proyectos, es la principal motivación de esta revisión

de la literatura. El aspecto principal en el que se enfoca este estado del arte, es la búsqueda de técnicas que apliquen la ingeniería inversa a interfaces gráficas de usuario con el fin de capturar requerimientos de software, lo cual está relacionado fuertemente a la comprensión de las aplicaciones, recuperación y reutilización del concepto. Lo anterior, con el fin de determinar los avances existentes y relacionados con esta área hasta el momento.

4.3.1 Metodología para la construcción del Estado del Arte

La metodología usada para estructurar esta revisión se basa inicialmente en las tres primeras etapas de Methontology [22] que determinan el alcance y dan estructura a la revisión de manera similar a como se inicia y determina el alcance de una ontología. En este caso estas etapas permiten delimitar el alcance de la revisión. La etapa de enumeración de términos es iterativa y va acompañada de la continua búsqueda de literatura, al finalizar estas etapas se procede al registro escrito, las correcciones y la divulgación de los resultados. Ver figura 5.

Figura 5. Diagrama Metodológico Estado del Arte



4.3.1.1 ¿Qué es una Ontología?

Una ontología es una descripción explícita y formal de los conceptos en un dominio de discurso. Se conforma de clases (algunas veces llamadas conceptos), propiedades de cada concepto que describen diferentes características y atributos de los conceptos (estas propiedades son llamadas ranuras, roles o simplemente propiedades); y también consta de restricciones sobre las ranuras o facetas (llamadas restricciones de rol). Una ontología junto con un conjunto de instancias de las clases constituye una base de conocimiento [23].

4.3.1.2 Descripción breve Methontology

Methontology es un enfoque iterativo para el desarrollo de una ontología. Fue desarrollado en el grupo de ingeniería ontológica en la Universidad Politécnica de Madrid, permite la construcción de ontologías al nivel de conocimiento y tiene sus

raíces en las principales actividades identificadas por el Instituto de Ingeniería Eléctrica y Electrónica, IEEE para el proceso de desarrollo de software. A continuación se describen las etapas de Methontology que se utilizan para el presente estado del arte.

Determinar el dominio y alcance de la ontología El desarrollo de una ontología se inicia determinando su dominio y alcance, para definir estos dos aspectos se debe dar respuesta a varias preguntas básicas. Algunas de ellas son:
¿Cuál es el dominio que la ontología cubrirá?
¿Para qué se va a usar la ontología?
¿Para qué tipos de preguntas la información de la ontología debe dar respuesta?
¿Quién va a usar y a mantener la ontología?
La respuesta a estas preguntas brinda un panorama inicial o punto de partida para el desarrollo de la ontología, sin embargo estas pueden cambiar durante el diseño y construcción de la ontología. Esto no es problema, siempre y cuando las nuevas respuestas limiten constantemente el alcance del modelo.
Además de las preguntas formuladas anteriormente es también conveniente formular preguntas de competencia, para ayudar a determinar el alcance de la ontología. Las preguntas de competencia son ese tipo de preguntas que la ontología debe ser capaz de responder, y servirán como guía de prueba más adelante, así se puede evaluar si la ontología contiene la suficiente información para dar respuesta a ellas si esas respuestas requieren un nivel de detalle o representación de un área en particular.

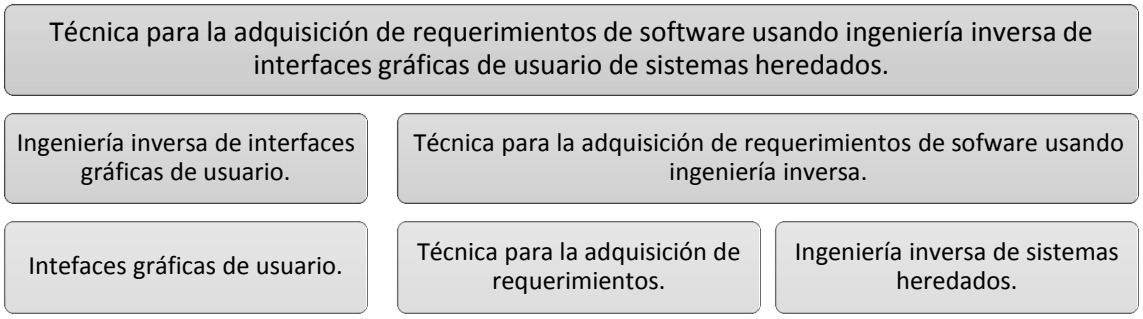
Enumerar términos importantes en la ontología Esta etapa consiste en escribir una lista de términos sobre los cuales se quiere hacer enunciados o ser explicados a algún usuario [23]. Este glosario de términos incluye todos los términos relevantes del dominio (conceptos, instancias, atributos, relaciones entre conceptos, etc.) sus descripciones en lenguaje natural y de ser posible sus sinónimos y acrónimos.

4.3.2 Construcción del Estado del Arte

Para el caso específico de este estado del arte la etapa preliminar al desarrollo de la metodología, es la búsqueda exhaustiva en la literatura sobre las técnicas existentes para la recolección de requerimientos de software mediante la ingeniería inversa de interfaces gráficas de usuario. Las primeras búsquedas realizadas en las bases de datos del IEEE y la Association for Computing Machinery, ACM que involucraban exactamente todos los términos de la temática planteada no arrojan resultados significativos al respecto, lo que sirve inicialmente para corroborar la hipótesis que es un área poco explorada donde se puede ubicar un trabajo de investigación de esta índole.

A continuación se procede a segmentar la búsqueda en las áreas principales que conforman la temática en general, figura

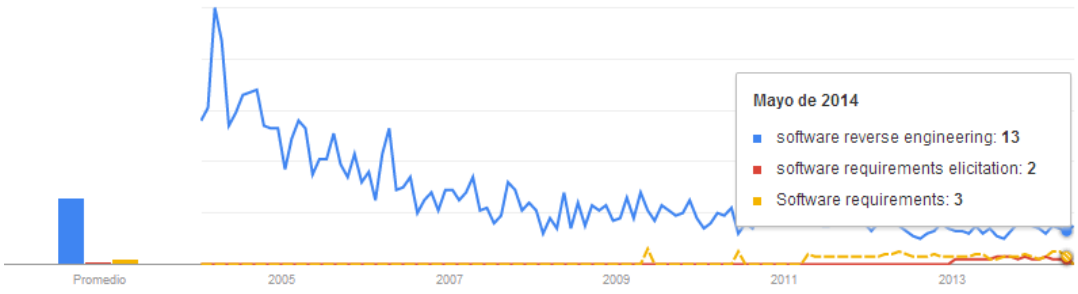
Figura 6. Segmentación de la temática para el proceso de búsqueda en Bases de Datos



Las búsquedas realizadas utilizando los términos de estas diferentes áreas permiten la recolección de artículos relacionados desde el año 1990 hasta la actualidad, se decide involucrar este amplio rango de tiempo en el análisis de la documentación ya que debido a la subdivisión hecha las temáticas se extienden a lo largo de estos veinticinco años. Además se detecta una aparición esporádica de las temáticas de interés en espacios de tiempo distantes. Sin embargo cabe aclarar que los artículos seleccionados se centran especialmente en los estudios encontrados al respecto desde el año 2000 hacia adelante.

Una revisión sobre las tendencias de búsqueda de los ejes temáticos centrales de esta investigación, muestra que el interés en temáticas como la ingeniería inversa ha decaído substancialmente en los últimos 10 años, mientras que la adquisición de requerimientos apenas está empezando a ser tema de búsqueda. Esto se ve reflejado en el número de resultados a lo largo de la línea de tiempo dada para este estado del arte, figura

Figura 7. Tendencias de Búsqueda



Fuente: Google Trends

Las etapas adaptadas de Methontology para estructurar la presente investigación son aquellas que están presentes en la concepción de la ontología. Estas son determinar el dominio y alcance de la ontología, realizar las preguntas de competencia y enumerar los términos importantes para la revisión. A continuación

se presentan los resultados de estas para la construcción del actual estado del arte.

4.3.2.1 Dominio y Alcance de la Ontología

En esta etapa se responden las preguntas básicas, las cuales permiten determinar el dominio y alcance, en la tabla 6 se muestran la respuestas para el presente caso.

Tabla 6. Respuestas a las preguntas básicas de Methontology

1. ¿Cuál es el dominio que la ontología cubrirá?
El dominio definido para esta ontología es la adquisición de requerimientos de software haciendo ingeniería inversa a las interfaces gráficas de usuario de aplicaciones heredadas.
2. ¿Para qué usaremos la ontología?
La presente ontología tiene diversos propósitos, el primero de ellos y más importante es determinar el estado del arte del dominio establecido y con esto demostrar la existencia de un nicho investigativo. Generando una base de conocimiento en este dominio que pueda ser asequible a otros investigadores interesados en el área.
3. ¿Para qué tipos de preguntas la información en la ontología deberá proveer respuestas?
• ¿Qué métodos o técnicas existen? • ¿Cuáles herramientas existen? • ¿Cuáles son los estudios más sobresalientes? • ¿Cuándo se desarrollaron? • ¿En qué consisten? • ¿Cómo usan la ingeniería inversa? • ¿Qué obtienen como resultado?
4. ¿Quién usará y mantendrá la ontología?
Inicialmente es para determinar un nicho investigativo para el presente trabajo de investigación, sin embargo queda como un producto disponible para todos los investigadores interesados en el área.

4.3.2.2 Preguntas de Competencia

Además de las preguntas formuladas anteriormente es también conveniente formular preguntas de competencia, para ayudar a determinar el alcance de la ontología. Las preguntas de competencia son ese tipo de preguntas que la ontología debe ser capaz de responder, y servirán como guía de prueba más adelante, así se puede evaluar si la ontología contiene la suficiente información para dar respuesta a ellas. En la tabla () se observan las preguntas de competencia para este caso.

Tabla 7. Preguntas de Competencia

¿Qué es ingeniería inversa?
¿Qué es adquisición de requerimientos?
¿A qué tipo de software se pueden aplicar las técnicas encontradas?
¿Alguna técnica utiliza un análisis que involucre únicamente la parte visible de la GUI y de la forma en que esta interactúa con el usuario?
¿Cuándo se mencionó por primera vez la posibilidad de utilizar la ingeniería inversa de GUIs para capturar requerimientos de software?

¿Cuáles técnicas se puede reconocer, que utilicen el análisis de aplicaciones heredadas para recuperar requerimientos o consideraciones de diseño?

4.3.2.3 Enumeración de términos importantes para la Ontología

La tercera etapa, es enumerar los términos importantes en la ontología. Esta fase se enfoca en encontrar los términos sobre los cuales se quiere hablar en la revisión y qué propiedades tienen estos términos. A continuación se muestra un extracto de esta lista de términos, cómo se clasificó cronológicamente y por documento consultado.

Tabla 8. Extracto lista de términos Ontología[6]

Nombre Documento	Año	Términos en inglés	Términos en español
Reverse Engineering and Design: A Taxonomy.	**1990**	Forward Engineering	Ingeniería Directa
Chikofsky, Elliot		Reverse engineering	Ingeniería Inversa
Cross, James		Design Recovery	Recuperación del Diseño
		Restructuring	Reestructuración
		Reengineering	Reingeniería
		Redocumentation	Re-documentación
		Specification language	Lenguaje de especificación
Reverse Engineering of user interfaces	**1993**	Method	Método
Merlo, E		Structural Representation	Representación Estructural
Girard, J F		Behavioural Representation	Representación Comportamental
Kontogiannisl, I		Lifespan	Vida útil
Panangaden, P		Reengineering UI	Reingeniería de la IU
Mori, R De		AUIDL*	AUIDL*
		User Actions	Acciones de Usuario
		System Responses	Respuestas del sistema
		Mapping out the behaviour	Mapear el comportamiento
		Process-Algebraic	Proceso algebraico
		Class Hierarchy based on Motif	Jerarquía de clases basada en Motif
		List of Equations	Lista de Ecuaciones

*AUIDL (Abstract User Interface Design Language)

4.3.2.4 Registro resumido de la información

Los resultados obtenidos para esta revisión de la literatura relacionada con la recuperación de requerimientos de software a partir de ingeniería inversa de interfaces gráficas de usuario se presentan a continuación. Estos resultados se presentan agrupados en etapas de tiempo las cuales dejan entrever las tendencias y enfoques. La primera sección comprende los años entre 1990 hasta el año 1999, la segunda sección se centra en los años 2000 al 2009 y finalmente la tercera parte cubre el intervalo de tiempo entre el año 2010 al 2013.

[6] La lista completa de términos se encuentra disponible en el Anexo B

Figura 8. Segmentación cronológica revisión de la literatura

PRIMERA ETAPA 1990 – 1999 Al inicio de este periodo, E. Chicofsky y J. Cross dedican una publicación a establecer de manera más clara los principios taxonómicos de la ingeniería inversa, que para la época aun no podía aplicarse exitosamente debido a la confusión que existía en lo referente a la terminología usada en campos técnicos y comerciales. En su documento definen seis términos muy involucrados en la comprensión y entendimiento del software, estando el término ingeniería inversa claramente definido como: “La ingeniería inversa es el proceso de analizar un sistema con el fin de identificar sus componentes y las interrelaciones entre ellos, y de crear una representación del sistema en otra forma o en un nivel más alto de abstracción.” [24], este artículo es la base para la gran mayoría de estudios citados en el presente análisis y el punto de partida para cualquier estudio referente la comprensión y entendimiento de un sistema heredado.

En años posteriores a la publicación de E. Chicofsky y J. Cross se presentan interesantes enfoques relacionados con la ingeniería inversa de aplicaciones y de sus interfaces de usuario. No siendo precisamente la década de los noventa el fuerte de la ingeniería inversa de interfaces gráficas de usuario, sin embargo se presenta un resumen de las tendencias dadas, relacionadas con este tema.

Entre los primeros enfoques en la década de los noventa, se encuentran los estudios como *Reverse engineering of user interfaces* de E. Merlo et al. [25]. Y *Reengineering user interfaces* también de E. Merlo [26], los cuales presentan un método para aplicar ingeniería inversa a las interfaces de usuario, donde se obtiene una representación tanto estructural como comportamental de la interfaz, con el fin de aplicar reingeniería a dicha interfaz y alargar el tiempo de vida del sistema. En cuanto al comportamiento la idea es describir los aspectos dinámicos de la interfaz, representando el comportamiento de la misma (la interacción entre sus componentes) a través de un proceso algebraico. Todo esto es automatizado mediante la implementación del lenguaje AUIDL, Abstract User Description Language. Este enfoque concluye que es posible generar especificaciones para interfaces gráficas de usuario simples a partir del análisis de interfaces de usuario

heredadas las cuales son orientadas a caracteres, y que trabajos futuros pueden lograr aplicar este método a interfaces gráficas de usuario.

Por lo anterior, es evidente, que para esta época aplicar ingeniería inversa a una interfaz gráfica de usuario no se considera una opción tan viable, ya que el desarrollo de los sistemas se encuentra aún en una transición entre interfaces de usuario, orientadas a caracteres e interfaces gráficas, sin embargo cabe resaltar que ya se considera seriamente el análisis de dichas interfaces para que esta transición sea más fácil de abordar.

Por otra parte, para este periodo de tiempo hay iniciativas interesantes en cuanto a la recuperación de requerimientos de software centradas especialmente en el análisis de los datos, además de diversos aspectos de las aplicaciones heredadas. Estudios como el de W. Cohen [27] proponen recuperar especificaciones a través del uso de técnicas de aprendizaje automatizado sobre las bases de datos que hacen parte de grandes sistemas heredados. Apoyando estas iniciativas se desarrollan trabajos relacionados con la implementación de aplicaciones que automatizan el proceso de ingeniería inversa de bases de datos [28].

De manera similar se encuentra el trabajo *Capturing Requiremens for Legacy Systems* [29] el cual presenta un enfoque para la recuperación de requerimientos de sistemas heredados, soportado por un proceso denominado Software Cost Reduction (SCR), lo interesante de este documento y que es importante citar, es que se propone enfáticamente que recuperar requerimientos a partir de código fuente es un proceso muy difícil y que somete al ingeniero a un proceso de especulación en cuanto a reconocer las decisiones que hizo el diseñador originalmente.

A partir de la segunda mitad de la década de los noventa se hizo evidente un creciente interés por reversar las antiguas interfaces de usuario con el fin de migrar o realizar mantenimiento de viejas aplicaciones de una manera más sencilla. Uno de estos trabajos es el presentado por M.M. Moore [30], publicado en el año 1996, la autora aborda la idea de construir un modelo de la funcionalidad de la interfaz, a partir del análisis estático y dinámico del código fuente y desarrollar una técnica que permita automatizar este proceso.

Para finales de los años 90, se encuentra un mayor número de trabajos que proponen técnicas para la recuperación de elementos de diseño desde el análisis de sistemas heredados. En *Reverse Engineering by Mining Dynamic Repositories* [31] se presenta el rescate de información junto con técnicas de minería del conocimiento como opción para aplicar ingeniería inversa a sistemas heredados. La diferencia fundamental es la visión basada en repositorios que permiten al programador no solo entender el modelo que está recuperando sino que también encuentra una descripción de cómo estos modelos han sido creados.

En el último año de la década se ubican trabajos de vital importancia por la relación con la temática de interés, el primero de estos es el de T. Richner y S. Ducasse *Recovering high-level views of object-oriented applications from static and dynamic information* [32], en este se expone un enfoque dedicado a la recuperación de la documentación arquitectónica a partir del código fuente. Aplicando para esto análisis estático y dinámico del código, con el fin de encontrar un modelo para representar programas. Finalmente también para el mismo año *Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour Environment Observation of Human - Machine Interactions* [10] exhibe un enfoque dirigido a la recuperación de requerimientos de sistemas heredados, analizando y modelando su comportamiento. La técnica denominada AMBOLS (Analysing and Modelling the Behavior of Legacy Systems), se fundamenta en tres etapas principales, captura del comportamiento, modelamiento del comportamiento dinámico y finalmente la derivación de requerimientos. En este trabajo se resalta la importancia de recuperar dichos requerimientos para entender a los interesados del sistema, obtener una imagen general de la funcionalidad, comprender la intención de diseño original y tener conocimiento sobre la interacción de usuario con el sistema.

SEGUNDA ETAPA 2000 – 2009 En consecuencia al auge de la ingeniería inversa y aplicaciones relacionadas con el análisis y comprensión de aplicaciones, las evidencias encontradas en esta segunda etapa se presentan divididas en dos periodos cronológicos y consecutivos.

Periodo 2000 – 2005 Para esta segunda etapa, los trabajos hallados en las bases de datos seleccionadas presentan resultados mucho más cercanos a la idea de aplicar ingeniería inversa a las interfaces de usuario de sistemas heredados, este es el caso de *Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach* [33], este documento publicado en el año de 1999 se considera dentro de esta etapa debido a la importante relación con otros trabajos posteriores a esta fecha, los cuales exponen diversas aplicaciones del método presentado.

El procedimiento para la ingeniería inversa de las interfaces de usuario expuesto en [33] se apoya en dos actividades fundamentales, la primera es el mapeo de la interfaz y la segunda el modelamiento de la tarea del dominio. El método de ingeniería inversa propuesto se implementa en el ambiente CelLEST, desarrollado especialmente para este propósito. Durante la primera mitad de la década del 2000 se presentaron varios trabajos estrechamente relacionados con CelLEST. En el primero de estos trabajos denominado *Modeling the System-User Dialog Using Interaction Traces* [34] se detalla LeNDI (LEgacy Navigation Domain Identifier), la herramienta prototipo para la ingeniería inversa de interfaces de usuario heredadas. LeNDI apoya la etapa de ingeniería inversa de CelLEST, no requiere análisis de código heredado y en lugar de esto se basa en el registro de las

trazas[7] de la interacción del usuario con el sistema. Dando continuidad a lo propuesto en [33] y [34], en *Dynamic Analysis for Reverse Engineering and Program Understanding* [35] se presenta el análisis dinámico como una opción formal y válida para la ingeniería inversa y el entendimiento de aplicaciones, enfocándose principalmente en el análisis del comportamiento del sistema heredado con el fin de comprender los procesos del mismo y sus usos, dando especial importancia a la ingeniería inversa de las aplicaciones y la comprensión de los artefactos de software. Derivado del conocimiento del proceso obtenido, en [36] se tiene como propósito recuperar los usos que los usuarios están dando actualmente al sistema, no se intenta recuperar los requerimientos considerados en la etapa de diseño del sistema heredado.

También para este primer periodo de la década del 2000 en *GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing* [37] hay una propuesta para aplicar ingeniería inversa a interfaces gráficas de usuario con el fin de realizar pruebas automáticas. El modelo obtenido se representa a través de arreglos que muestran la estructura que forman las ventanas de la aplicación y la relación jerárquica que hay entre ellas. Este enfoque no se extiende al análisis de código sino que dirige su atención al reconocimiento de los componentes y las propiedades de los mismos y los valores dados a estas propiedades en una determinada GUI.
Finalizando este periodo de tiempo hacia el año 2005 Y. Yu, Y. Wang, y J. Mylopoulos formulan en su trabajo *Reverse Engineering Goal Models From Legacy Code* [38] la refactorización de código heredado para extraer modelos de las metas del software, la idea presentada de manera general es reconocer la intención del software. A pesar de la importancia de reconocer la intención de un sistema en la captura de requerimientos el proceso se basa únicamente en el análisis de código.

Periodo 2006 – 2009 En este periodo el uso de las aplicaciones web y las llamadas aplicaciones enriquecidas de internet (Rich Internet Applications, RIAs), está totalmente establecido, por esto es notable como las técnicas se orientan especialmente a este tipo de aplicaciones software. Sin embargo hay un fortalecimiento de las técnicas enfocadas al análisis del código fuente, y aunque se busque reversar GUIs casi siempre se recurre al código como opción para aplicar las técnicas de ingeniería inversa.

Iniciando este periodo de tiempo se presenta un interesante trabajo denominado *Application Modelling Using Reverse Engineering Techniques* [39] dirigido al análisis de código de aplicaciones web desarrolladas en ASP.NET[8], con el fin de extraer un modelo de la aplicación. La metodología propuesta cuenta con una

[7] Secuencia de capturas de pantalla que son intercaladas por las acciones de usuario desarrolladas.
[8] ASP.NET es un producto de Microsoft Corporation

herramienta software que realiza el análisis estático del código contenido de los archivos aspx y de los archivos cs que contienen la lógica de la aplicación. Al final del proceso se obtiene una representación del modelo conceptual de la aplicación usando la notación WebML[9].

Para esta época la ingeniería inversa es una herramienta especialmente enfocada al mantenimiento y esto se refleja en trabajos como el de [40] el cual tiene un enfoque orientado especialmente a tareas de mantenimiento tales como la migración de una interfaz gráfica hacia los llamados constructores de GUI y documentación de la GUI. El objetivo principal es mediante el análisis estático del código fuente, extraer las jerarquías de los componentes que forman las ventanas de los programas junto con los atributos de dichos componentes y los manejadores de eventos. Este trabajo se realiza en código escrito en C y C++.

Continuando con la tendencia de análisis de código de aplicaciones web, en *Reverse Engineering Finite State Machines From Rich Internet Aplications* [41] se sigue un enfoque de ingeniería inversa que pretende abstraer máquinas de estado finitas (*Finite State Machines, FSM)*, las cuales representan el comportamiento ofrecido por una RIA hacia el lado del cliente. El proceso consiste primero en la extracción, la cual se realiza por medio del análisis dinámico del código fuente, y registra en diagramas de estado la actividad hecha por la RIA ejecutándose en un ambiente controlado para una sesión de usuario. La segunda actividad o abstracción, inicia con los datos recolectados en la etapa de extracción y tiene tres pasos fundamentales: la generación de un grafo de transición (Transition Graph, por sus siglas en inglés), agrupación y asignación de concepto. Al final del segundo paso se obtiene un grafo de transición simplificado, el último paso toma los nodos del TG y asume cada uno de estos como estados de la máquina de estados, los estados deben ser validados por un ingeniero de software. Todo esto se logra junto con una herramienta implementada especialmente para la automatización del proceso.
En general para el final de esta década la ingeniería inversa se soporta completamente en el análisis estático y dinámico del código fuente, incluso para los enfoques que proponen el análisis de las interfaces gráficas. Ya que la interfaz se analiza desde su código. Además aunque no centrados en la interfaz gráfica o en la recuperación de requerimientos trabajos como [42] están dirigidos a lenguajes de programación orientados a objetos y otros a esquemas XML [43].

TERCERA ETAPA 2010 – 2013 En esta última etapa se encuentran trabajos con diversos enfoques, se fortalece la tendencia que viene desde la etapa anterior sobre la aplicación de ingeniería inversa a las RIAs y se presentan novedosos procesos orientados a las GUIs construidas usando aplicaciones RAD (*Rapid Appliation Development*). Además durante este periodo y debido al auge de las

[9] WebML es una notación visual utilizada en el diseño de aplicaciones web complejas y con un uso intensivo de datos.

aplicaciones móviles se inicia el interés por encontrar técnicas que faciliten la migración a estas nuevas tecnologías.

Por lo tanto es de interés para los trabajos de esta época reversar GUIs y encontrar modelos independientes de la tecnología, que permitan hacer más fácilmente mantenimiento y migración de las interfaces. Un trabajo de 2010 muy relacionado con lo anterior es el desarrollado por [44] en el cual se expone un método de ingeniería inversa de interfaces gráficas de usuario que hayan sido implementadas usando un constructor de GUIs, es decir arrastrando y arrojando componentes desde una paleta. El objetivo principal de este enfoque es el de llegar al modelo de una interfaz de usuario concreta (CUI, Concrete User Interface) el cual es un representación independiente de la tecnología.

En trabajos posteriores y dando continuación al trabajo hecho sobre ingeniería inversa de RIAs en [41] y el objetivo de encontrar modelos independientes de la tecnología, el artículo titulado *An Iterative Approach For the Reverse Engineering of Rich Internet Application User Interfaces* [45] propone un método iterativo que utiliza las máquinas de estado finitas como representación del modelo del comportamiento de la interfaz gráfica de usuario de una RIA. Este método se compone de cuatro pasos fundamentales y un criterio de salida. Los cuatro pasos básicos son Interacción de Usuario, Extracción, Abstracción y Asignación de Concepto, estas tareas son realizadas a través de análisis dinámico en un ambiente integrado de ingeniería inversa denominado CReRIA[10].
Para el 2011 se destaca como el auge de las aplicaciones móviles también influye en el interés de migrar aplicaciones comúnmente web o de escritorio a plataformas móviles, la ingeniería inversa de la interfaces gráficas se presenta como una poderosa opción ya que para un ambiente móvil el despliegue de la interfaz cambia substancialmente, sin embargo se quieren conservar sus características básicas de estructura y comportamiento. El enfoque presentado en *Reverse Engineering User Interfaces to Facilitate Porting to and across Mobile Devices and Platforms* [46] ilustra el proceso de obtener un modelo general de la GUI, el modelo surge a partir del análisis de la representación en tiempo de ejecución de la interfaz lo que evita el análisis del código fuente.

Finalizando el periodo de tiempo acotado para el presente estado del arte, se reconoce que los enfoques dados a la ingeniería inversa de GUIs se mantienen orientados hacia la obtención de modelos de la estructura y comportamiento de la interfaz. Se populariza el uso de máquinas de estado finitas para representar estos modelos y se buscan herramientas y métodos que mejoren los resultados de estos modelos, en *GUI Reverse Engineering with Machine Learning* [47] se propone una mejora para estos modelos a través de la programación lógica inductiva, esta identifica y resuelve las condiciones en las cuales ocurren ambigüedades en el

[10] No se encuentra una denominación para sus siglas.

modelo, discriminando dos situaciones y determinando cual es el siguiente estado para la FSM.

Hacia el 2013 los enfoques se siguen centrando en el análisis dinámico de las aplicaciones para obtener modelos del comportamiento de la GUI, ya que como está comprobado para ese momento, los enfoques de análisis estático del código fuente son muy eficientes para extraer información sobre la estructura interna del sistema y las dependencias entre elementos estructurales, pero presentan falencias a la hora de recuperar modelos de comportamiento de GUIs. A su vez el análisis dinámico se presenta como la única solución viable que no requiere del código fuente para extraer el comportamiento de la GUI, sin embargo se reconoce que es una alternativa más difícil de automatizar y que además de requerir la herramienta que automáticamente analiza la aplicación, necesita la inserción de datos coherentes a la GUI de manera manual [48].

En general y de acuerdo a lo encontrado en la revisión de la documentación, es notorio que el interés de reversar GUIs se concentra en la obtención de modelos que permitan la realización de migración y pruebas. En cuanto al interés de aplicar ingeniería inversa a las interfaces gráficas de usuario para encontrar requerimientos de software, se encuentra una interesante iniciativa hacia el año 2007, este pretende insertar la ingeniería inversa de aplicaciones heredadas dentro del ciclo de vida del software, integrando los requerimientos clásicos con requerimientos obtenidos por ingeniería inversa, y presenta la posibilidad de obtener requerimientos a partir de los modelos gráficos proporcionados por procesos de ingeniería inversa. [9] Sin embargo y de acuerdo al estudio hecho en el presente estado del arte se obtiene como resultado que esta iniciativa no tuvo incidencias en trabajos posteriores y que es un amplio campo investigativo disponible para nuevas ideas y enfoques del uso de la ingeniería inversa de GUIs.

5. DESARROLLO DE LA TÉCNICA

De acuerdo al modelo metodológico propuesto, la etapa desarrollo de este proyecto está dirigida por el estadio de investigación proyectiva. Este tipo de investigación propone la construcción de un producto o solución al problema planteado, fruto de los resultados obtenidos en estadios previos de la investigación.

La solución propuesta para el problema de investigación dado es una técnica que se apoye en el análisis inverso de las interfaces gráficas de usuario de aplicaciones heredadas. Para tal fin la investigación proyectiva plantea el uso de ciclos iterativos, mediante estos ciclos van logrando versiones de la técnica y las respectivas mejoras a cada una de dichas versiones.

A continuación se presentan algunas de las consideraciones de diseño y los resultados más importantes obtenidos en las iteraciones hechas en la construcción de la solución propuesta. En este caso se realizaron múltiples cambios que fueron agregando características a la técnica de captura de requerimientos, sin embargo para ser presentadas en este documento se agrupan en cuatro grandes iteraciones o ciclos de construcción, ya que son los momentos donde la técnica muestra mayores cambios.

5.1 ANÁLISIS PRELIMINAR

Teniendo en cuenta lo establecido por las definiciones formales de ingeniería inversa y a los hallazgos evidenciados en el estado del arte, es claro que la fuente principal y fundamental que se ha considerado a la hora de aplicar ingeniería inversa a un producto de software es el código fuente. Sin embargo y citando lo enunciado en el marco teórico “el objetivo de la ingeniería inversa es obtener la descripción más confiable tanto del comportamiento del sistema como de la estructura del negocio y sus reglas”, es entonces claro que hasta el momento los enfoques propuestos se han centrado hacia el descubrimiento del sistema, la estructura del mismo y su comportamiento, lo que ha desembocado en múltiples estudios orientados especialmente al análisis estático y dinámico del código fuente y la aplicación. Estos enfoques requieren especial esfuerzo por la complejidad de las aplicaciones que lo realizan, el hardware requerido para soportar estas aplicaciones y la especialización de las personas que interpretan los resultados de la ingeniería inversa.

De lo anterior surgen dos preguntas importantes ¿qué otras fuentes pueden estar disponibles para la ingeniería inversa? Y ¿cómo pueden usarse esas fuentes para elucidar las necesidades de información derivadas del uso de la aplicación de software por parte de los clientes y usuarios, y entender mejor la estructura del negocio y sus reglas?

Estas dos preguntas son la base fundamental sobre la cual se plantea la técnica descrita en este trabajo de investigación. La respuesta a la primera se orienta hacia el elemento de interacción usuario - aplicación por excelencia, la interfaz gráfica de usuario GUI. La segunda respuesta conduce a la concepción de un método organizado que permita estudiar la GUI para extraer suficiente conocimiento del dominio de aplicación como para proveer elementos suficientes para proceder con el desarrollo de una nueva aplicación o la realización de reingeniería a aquella en estudio.

Al mismo tiempo que se considera lo anteriormente descrito, se plantea la importancia de obtener una técnica que cuente con dos características fundamentales, la primera que sea sencilla de aplicar, y la segunda que los resultados obtenidos sean comprensibles fácilmente. La idea es no requerir de expertos en confusos algoritmos de ingeniería inversa ni expertos en interpretar modelos arquitecturales complejos de software. El objetivo es que cualquier desarrollador pueda disponer de un conjunto de heurísticas que le permitan aprender de lo existente y comprender más fácilmente el dominio al que se enfrenta en un nuevo desarrollo o en un proceso de reingeniería.

5.2 PRIMERA ITERACIÓN

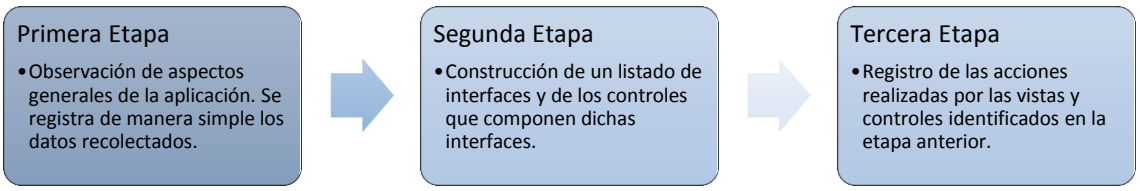
Desde el inicio de este proyecto se tienen muy en cuenta los siguientes principios, los cuales dirigen el desarrollo de la presente técnica:

- Ninguna intervención ni observación del código fuente.
- Poca o ninguna interacción con los usuarios.
- Ninguna interacción con las personas involucradas en el diseño.
- Inexistencia de documentación sobre el diseño o de las modificaciones hechas a través del tiempo en la aplicación.
- Una aplicación ejecutable que está cumpliendo parcialmente con las necesidades del negocio y que es utilizada, pero que ya requiere ser reemplazada por una que cumpla con esas necesidades ya satisfechas y con las más nuevas, sean funcionales o de tecnología. Este tipo de aplicaciones por lo general requieren ser reemplazadas por razones como:
    - Necesitan nuevas funcionalidades.
    - Tienen funcionalidades que no son necesarias.
    - Necesitan ampliar el número de plataformas donde son ejecutadas.
    - Evolucionar de ambientes de escritorio a ambientes web o móviles.
    - Mejorar en extensibilidad.
    - Mejorar la usabilidad.
- Una aplicación ejecutable de la cual se tiene conocimiento se usa de manera efectiva en un dominio determinado y de la cual se quiere conocer que requerimientos puede satisfacer.

- Obtener consideraciones hechas en el diseño en un formato natural y de fácil entendimiento para los interesados. Dentro de los interesados se contemplan los usuarios, quienes usan el sistema actualmente y en un futuro usarían las nuevas versiones del mismo o el sistema que lo reemplace. Otro tipo de interesados son los clientes, quienes desean cambiar el sistema actual pero no perder ninguno de los beneficios que les ofrece. Los diseñadores y desarrolladores quienes estarán al frente de la construcción de un nuevo sistema que cuente con los beneficios del actual, elimine sus debilidades y llene sus falencias.

La primera versión de esta técnica no cuenta con muchos elementos que permitan generar como resultado requerimientos de software estructurados de manera formal. Es más un conjunto de consideraciones previas, hechas para determinar a grandes rasgos los principales elementos que delimitan la técnica a desarrollar. En la figura se muestra el esquema de bloques planteado para la primera iteración donde se enumeran de manera simple las ideas más importantes para cada una de estas etapas, partiendo del momento en el que el analista se enfrenta por primera vez a una aplicación desconocida y busca identificar en ella elementos claves de su diseño.

Figura 9. Esquema de bloques inicial para la técnica



Como primera etapa propuesta para la técnica se piensa en una observación y registro simple del tipo de aplicación (web o escritorio), el sistema operativo en el que se ejecuta, el navegador o navegadores en que funciona (si es web) y el ambiente de negocio en el que se usa.

A partir de esta observación inicial, se reconocen también las interfaces alcanzadas por el usuario durante la ejecución, se registra el nombre de la interfaz en una lista, y la forma en la cual se alcanza dicha vista.

En este caso pensando en el software como algo intangible, pero que de alguna manera el usuario percibe como algo casi viviente, cabe la pregunta ¿cómo el usuario percibe ese objeto tan conceptual, tan intangible, que simplemente reside en los estados lógicos de un dispositivo electrónico? Simple, a través de su rostro y ¿cuál es el rostro de una aplicación? Su interfaz.

Entonces de alguna manera esa interfaz es el límite desde el punto de vista del usuario de ese objeto llamado software. Y es el límite mediante el cual lo conoce e

interactúa con el diariamente como herramienta de trabajo. Desde ese límite conocido inicia la segunda etapa planteada por la técnica en construcción. La cual busca encontrar las funciones que satisface la aplicación al usuario.

La segunda etapa pensada para esta versión es el primer acercamiento al análisis de la interfaz gráfica, aquí es donde gracias al principio básico de la ingeniería inversa es posible obtener a partir de un producto terminado información sobre las consideraciones de diseño hechas por sus constructores. En este caso el producto terminado al cual el analista tiene mayor contacto es el software en ejecución, entonces el analista debe interactuar y registrar de alguna forma los resultados de dicha interacción.

En este primer acercamiento la tercera etapa no se destaca muy claramente, simplemente se plantea como un registro de las acciones que el usuario realiza con los controles y en las interfaces identificadas.

5.3 SEGUNDA ITERACIÓN

La primera parte de esta segunda iteración se inicia con dos tareas importantes a desarrollar, la primera es darle un nombre a la técnica con el fin de referirse a la misma de una manera más sencilla y clara; la segunda es determinar el tipo de formato que deben tener los requerimientos obtenidos a través de esta técnica. Para la segunda parte de esta iteración se planea diseñar el formato que va a tener el listado de interfaces y el listado de controles de cada una de las interfaces, donde se realiza el registro de todos los componentes en interfaces disponibles para el análisis.

5.3.1 Primera Parte: Nombre para la técnica y formato de requerimientos

En cuanto a la primera tarea se resuelve dando un nombre relacionado a las siglas en inglés del objetivo que tiene este proyecto, el cual es aplicar ingeniería inversa para capturar requerimientos de software, entonces se obtiene: Reverse *Engineering For Requirements Elicitation ReFre* al final se decide agregar una “e” más, dando la forma final del nombre de la técnica ReFree, teniendo como resultado el efecto de la palabra libertad o independencia de manera implícita en el nombre y siendo una de las cosas que esta técnica busca dar al analista a la hora de aprender sobre un dominio determinado.

Para el desarrollo de la segunda tarea, la cual pretende determinar el formato de requerimientos a usar, se presenta inicialmente la idea de formular los requerimientos obtenidos en un modelo estándar de requerimientos, esto es a través de casos de uso. Sin embargo existen otra alternativa que se decide analizar y tener en cuenta, esta es el uso de Características.

Por lo anterior es necesario estudiar tanto casos de uso como características, para realizar esto se utilizó además del conocimiento derivado del marco teórico un interesante hallazgo el cual presenta un paralelo entre Características y Casos de Uso, tabla 9.

Tabla 9. Características vs. Casos de Uso

Enfoque de Características	Enfoque Casos de Uso
Se centra en características específicas de la aplicación, trata de tomar una pieza de funcionalidad que el usuario desea tenga la aplicación, y trabajar en dicha funcionalidad hasta que esté completa	Se enfoca en flujos específicos a través de la aplicación. Este enfoque toma una ruta completa a través de la aplicación, con un claro inicio y final e implementa esa ruta en el código.
Trabaja muy bien cuando se tienen múltiples características diferentes que no se interconectan en un todo.	Cuando el desarrollo enfocado hacia casos de uso e trabaja en un escenario completo.
Permite mostrar al usuario código funcional más rápidamente.	Permite mostrar al usuario piezas mayores de funcionalidad en casa etapa de desarrollo.
Es muy orientado a la funcionalidad	Es muy centrado al usuario. Se codificará para todas las diferentes formas en que un usuario utiliza el sistema.
Trabaja particularmente bien en sistemas con muchas piezas de funcionalidad desconectadas.	Trabaja bien para aplicaciones que tienen bastantes procesos en vez de piezas individuales de funcionalidad.

Fuente: Head First Object-Oriented Analysis and Design (Traducción Libre) [13].

Teniendo en cuenta que ReFree se propone como una técnica orientada hacia el análisis de una determinada aplicación sin interacción con el usuario, se reconoce que las Características, al ser especialmente orientadas a la aplicación, encajan como medios para la formalización de requerimientos.

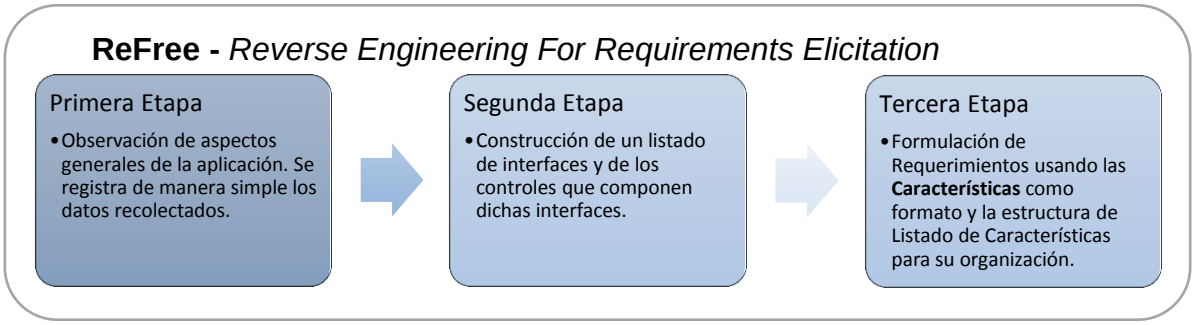
Los casos de uso por su parte se obtienen normalmente de la interacción con el usuario. Después de la aplicación de ReFree se puede aplicar la obtención de casos de uso, si se cuenta con la disponibilidad de tiempo de los usuarios, con el fin de completar los requerimientos, y si los conjuntos de características derivados no son estimados suficientes para el desarrollo de un proceso de reingeniería de la aplicación en cuestión.

Otro elemento importante en la elección de las características como formato de los requerimientos, es que, al tratar de recuperar consideraciones de diseño a partir de una aplicación heredada no es posible identificar flujos específicos a través de la aplicación, sino que a medida que se avance en el análisis de las distintas interfaces se recolectan piezas de funcionalidad separadas, que posiblemente no se puedan relacionar fácilmente.

El análisis de los dos enfoques permite definir las Características como el formato a usar en ReFree para derivar los requerimientos de software a partir de una

aplicación heredada, entonces el incipiente diagrama de bloques de la figura 9, evoluciona al diagrama presentado a continuación, ver figura 10.

Figura 10. Diagrama a bloques de ReFree evolución parcial segunda iteración



5.3.2 Segunda Parte: Diseño Listado de Interfaces y Controles

El listado de interfaces mencionado en la primera iteración evoluciona y se complemente con el denominado **Árbol de Interfaces**, y el listado de controles se propone como la **Matriz de Componentes**. Estos elementos son útiles en el registro de las interfaces disponibles para el análisis y los componentes de la interfaz gráfica que hacen parte de cada una de las interfaces disponibles.

El diseño de estos dos importantes elementos pertenecientes a la segunda etapa de ReFree, se basa tanto en conceptos y teorías propias de la ingeniería del software como experiencias compartidas entre el autor y el director de la presente investigación.

5.3.2.1 Árbol de Interfaces

El árbol de interfaces es de manera simplificada una lista de todas las interfaces a la que se puede tener acceso en la aplicación heredada, el objetivo principal de esta lista es registrar un inventario de todas las interfaces o vistas disponibles, además que se observe fácilmente la estructura general de navegación de la aplicación.

El árbol de interfaces debe mostrar claramente el título o nombre dado a la interfaz, un identificador numérico único y las interfaces por medio de las cuales se llega a determinada vista de la GUI. Este árbol puede construirse inicialmente como una estructura de listas sencillas con ayuda de una hoja de cálculo como Excel[11], incluso si la aplicación a simple vista no es muy extensa este árbol puede construirse a mano con lápiz y papel. Sin embargo se piensa en el uso de alguna herramienta software para que permita obtener un modelo más gráfico del árbol de interfaces.

[11] Microsoft Corporation

5.3.2.2 Matriz de Componentes

La matriz de componentes surge de la necesidad de registrar todos los controles que contienen cada una de las interfaces, ya que es a través de estos controles que el usuario logra realizar todas las actividades para la cuales está diseñado el software. La idea es registrar organizadamente cada control, y sus características fundamentales y visibles a simple vista. Entre las características consideradas en esta segunda iteración están:

- Tipo de control, ejemplo botón, caja de texto, botón de radio, etc.
- Acción que realiza, en esta parte se registra solo el verbo asociado a la acción, guardar, eliminar, salir.
- Elemento, es decir el elemento sobre el cual se realiza la acción registrada anteriormente.
- Relación, en este caso se busca si el control está asociado a otros controles dentro de la interfaz en alguna configuración específica. Es decir si pertenece a algún menú, barra de herramientas, etc.
- Función, se describe con más detalle la acción que se puede realizar dentro del software a través del uso de este control.

La idea principal es lograr encontrar las funcionalidades que tiene la aplicación a través de la observación y manipulación de los controles de la aplicación. Para luego traducir estas sencillas funciones en requerimientos funcionales expresados como características. Una primera versión de esta matriz se puede observar en la figura 11.

Figura 11. Primera versión Matriz de Componentes

Nombre de la Interfaz: Interfaz de Prueba				
Tipo de Componente	**Etiqueta**	**Elemento que Afecta**	**Relación**	**Función**
Botón	Eliminar	Cliente	Se encuentra en una barra de menú.	Borra el registro de un cliente de la lista.
Text Box	Nombre Cliente	Cliente	Grupo de cajas de texto en un formulario.	Permite ingresar el nombre del cliente.
Radio Button	...	...	...	...
...	...	...	...	...

5.4 TERCERA ITERACIÓN

En esta iteración, se formalizan aspectos como la observación inicial, perteneciente a la primera parte de la técnica, se agregan nuevos elementos al árbol de interfaces y la matriz de componentes. Y se reclasifican las actividades de la primera y segunda etapas de ReFree.

5.4.1 Primera etapa de ReFree: Inspección inicial aplicación heredada

ReFree entonces propone una inspección inicial sobre el ambiente en el que la aplicación heredada está siendo operada. Esta inspección determina parámetros generales sobre la aplicación. Hay dos tipos de información la cual se desea obtener en esta fase simple de reconocimiento, el primer tipo es la información relacionada con características técnicas de la aplicación heredada reconocibles a simple vista:

- Tipo de aplicación.
- Sistemas operativos sobre el que se ejecuta.
- Formatos disponibles para el análisis.
- Módulos Disponibles para el análisis.
- Registro simple de mapas de ejecución para cada módulo.

El segundo tipo de información a obtener en esta fase, depende de la oportunidad de entrevistar puede ser de manera informal a un individuo que tenga contacto con la aplicación o tal vez con el diseño. Esta es una situación poco probable, pero de ser el caso, el ingeniero de ReFree debe obtener tanta información como sea posible, los parámetros a registrar vía observación u entrevista son:

- Propósito u objetivo de la aplicación.
- Quienes son los usuarios de la aplicación.
- En qué forma usan la aplicación.
- Dónde se usa la aplicación.

La meta en esta etapa preliminar es identificar y registrar aspectos generales que puedan ser útiles en fases posteriores de ReFree. En la actividad final de esta etapa, la información obtenida de la ejecución de la aplicación es registrada, esto es la construcción del árbol de interfaces, el cual permite apreciar el tamaño de los diferentes módulos y el número de vistas con el cual se cuenta para el análisis de la aplicación.

5.4.1.1 Registro de Parámetro Iniciales

Se decide diseñar un formato simple que el analista ReFree debe completar, el formato es muy sencillo y sirve para que no se olvide registrar ninguna de la información relevante determinada para esta etapa. En la figura 12, se muestra el formato final para el desarrollo de esta actividad.

Figura 12. Formato parámetros iniciales

Formato Parámetros Iniciales	
Nombre de la aplicación:	**Fecha:**
Tipo de aplicación:	
S.O sobre el cual se ejecuta:	
Propósito u objetivo de la aplicación:*	
Quienes son los usuarios de la aplicación:*	
Forma en que se usa la aplicación:*	
Dónde se usa la aplicación:*	
Número de Módulos para analizar:**	**Número de Formatos para analizar:****
Observaciones:	

*Las respuestas a esta preguntas pueden darse a través de una observación simple o en el caso de ser posible a partir del diálogo con algún usuario o persona en contacto con la aplicación
*Esta información se completa después de construir el árbol de interfaces propuesto como segunda actividad de ReFree, en el caso de ser un segmento de la aplicación el que se va a analizar se recomienda especificarlo en el campo de observaciones

5.4.1.2 Revisión final Árbol de Interfaces

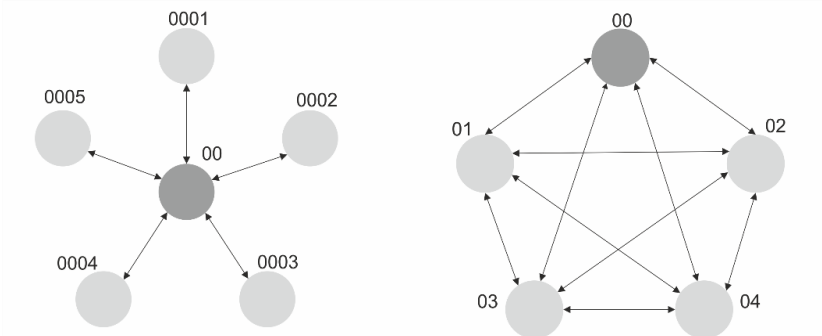
En cuanto al árbol de interfaces, se mantienen las ideas propuestas en la iteración anterior, el árbol debe presentar como características fundamentales: Servir de inventario de interfaces y mostrar la estructura del modelo de navegación general de la aplicación. Como este árbol debe ser a su vez un inventario se debe entonces tener para cada interfaz un identificador único o ID, que acompañe, como se había planteado en la segunda iteración el nombre de la interfaz y que señale la interfaz de la que se proviene y las que se pueden alcanzar desde la vista actual.

A continuación se presenta el modelo de identificadores propuesto, el cual se ha diseñado teniendo en cuenta los modelos de navegación estudiados en el marco teórico. Los identificadores aquí propuestos buscan entregar información útil al analista sobre la posición de la interfaz dentro del sistema, sin embargo no

pretenden ser camisa de fuerza para el analista ReFree el cual puede utilizarlos, modificarlos o simplemente generar su propio formato de identificadores.

- **Concentrador y Radios (Hub and Spoke)** Como se observa en la figura 13a, para este caso la asignación de los identificadores se hace fijando el número 00 a la pantalla inicial, y a las pantallas en los radios los números 0001, 0002, 0003,..,000n, donde 00 indica su vínculo inmediatamente contiguo a la pantalla de inicio y 01, 02, 03,…,0n el número de pantalla asignada.
- **Completamente Conectado (Fully Connected)** La asignación de identificadores en este modelo, donde todas las páginas se encuentran al mismo nivel de profundidad se hace de manera continua. Figura 13b.

Figura 13. Identificadores modelo de navegación Concentrador y Radios y, Completamente Conectado

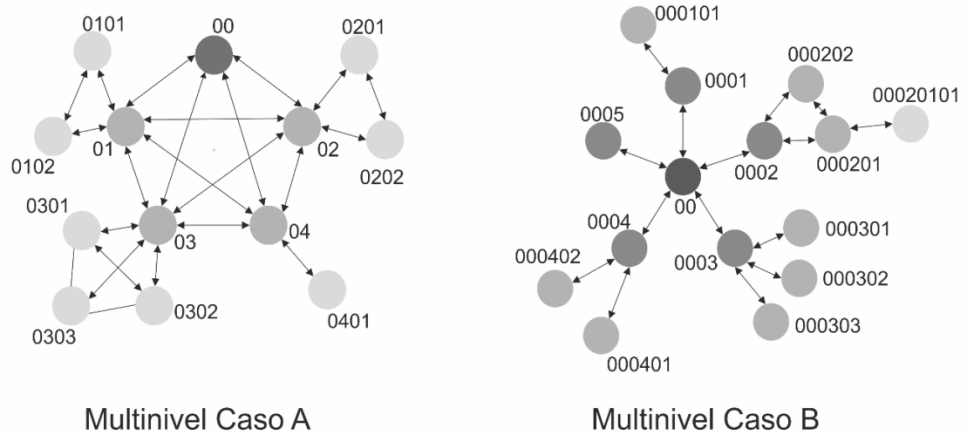


a. Identificadores modelo Concentrador y Radios

b. Identificadores modelo Totalmente Conectado

- **Multinivel** La asignación de identificadores en este modelo de navegación busca mostrar implícitamente dentro del código desde que pantalla se invoca otra interfaz. Inicia con el valor 00, asignado a la página de inicio o página principal y va combinando valores dependiendo desde donde se invoca cualquiera de las interfaces, puede considerarse una combinación de los dos modelos anteriores.

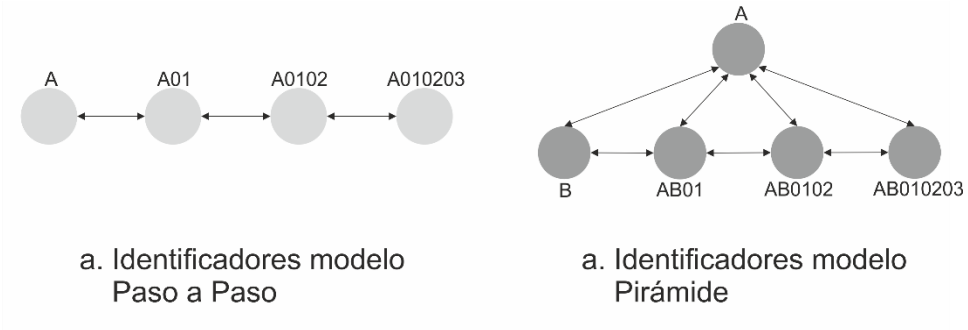
Figura 14. Identificadores modelo de navegación Multinivel (dos casos)



En los dos modelos de navegación a continuación, los identificadores propuestos varían parcialmente de los anteriores, sin embargo en este caso la inclusión de letras dentro de los códigos se hace con el fin de abreviar la longitud de los identificadores, a su vez se tiene en cuenta que las posibilidades en la navegación son más restrictivas y lo más importante es destacar el hecho de que son modelos de navegación secuencial.

- **Paso a Paso** En este modelo la primera página o página de origen se identifica con la letra A mayúscula, las demás páginas se enumeran de la forma expuesta en la figura ()a. De este modo se identifica que existe un solo origen y que las demás dependen de su página anterior para poder ser accedidas. Es decir son secuenciales.
- **Pirámide** De manera análoga la página de origen se denota con la letra A, solo que en este caso la página de origen accede a cualquiera de las páginas en la secuencia y viceversa, pero no es la primera página de la secuencia, a la primera página de la secuencia se le denota entonces con la letra B. Figura ()b.

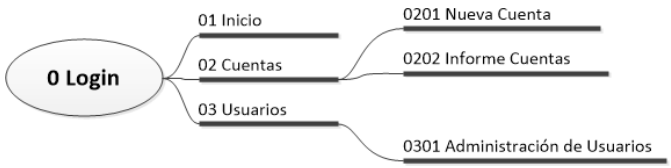
Figura 15. Identificadores modelo de navegación Paso a Paso y Pirámide



Además de la asignación de los códigos en el árbol de interfaces, se debe registrar el nombre o título que tenga la interfaz o página registrada. El uso de una

herramienta software que permita crear el listado mostrando gráficamente, permite observar la claramente la estructura que puede llegar a tener la aplicación, a continuación se presenta un ejemplo genérico desarrollado con ayuda del software Microsoft Visio[12]. Figura 16.

Figura 16. Ejemplo Árbol de Interfaces



5.4.1.3 Segunda revisión Matriz de Componentes

En esta segunda revisión que se hace a la matriz de componentes, se tienen en cuenta los nuevos elementos obtenidos en el análisis preliminar de la aplicación, en especial el árbol de interfaces, el cual entrega un identificador único de cada una de las interfaces que van a ser analizadas. Este dato es agregado a la matriz de componentes, se añade también la columna e imagen componente o control que se está registrando, el cual ayuda a su fácil identificación dentro de la interfaz en estudio. Además no todos los controles, en especialmente los botones tienen una etiqueta escrita, algunas veces estos contienen imágenes que indican su propósito. En la figura 17 se puede observar la segunda versión de la matriz de componentes.

Figura 17. Segunda versión Matriz de Componentes

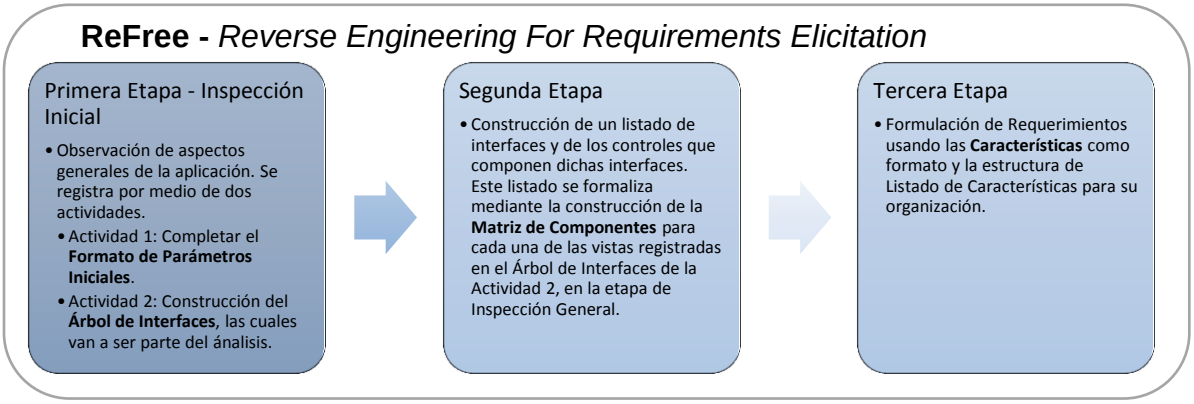
ID Interfaz: 00010101		**Nombre de la Interfaz:** Interfaz de Prueba			
Tipo de Componente	**Imagen Componente**	**Etiqueta**	**Elemento que Afecta**	**Relación**	**Función**
Botón	Eliminar	Eliminar	Cliente	Se encuentra en una barra de menú.	Borra el registro de un cliente de la lista.
Text Box	Nombre Cliente	Nombre	Cliente	Grupo de cajas de texto en un formulario.	Permite ingresar el nombre del cliente.
Radio Button	Radio Button	...	...	...	...
...		...	...	...	...

Hasta este punto el diagrama de bloques de ReFree resulta en lo siguiente. Primera etapa o Inspección Inicial, en esta parte se realiza la observación general de la aplicación, sin embargo se formaliza el registro de los datos obtenidos por medio del Formato de Parámetros Iniciales y el Árbol de Interfaces. La segunda etapa consiste en examinar cada una de esas interfaces y "descomponerlas" registrando todos los elementos que las conforman, estos elementos son los

[12] Microsoft Corporation.

controles o componentes de interfaz de usuario. La tercera etapa hasta el momento no ha sufrido más cambios, estos se exponen en la siguiente iteración.

Figura 18. Diagrama de bloques ReFree, evolución parcial tercera iteración



5.5 CUARTA ITERACIÓN

En esta cuarta iteración suceden procesos relevantes dentro del desarrollo de ReFree, especialmente en lo relacionado con la generación de requerimientos y la forma en cómo se unen los elementos de la Matriz de Componentes al esquema de características presentado en el marco conceptual de este trabajo.

Lo primero a considerar es el estudio desarrollado por Russell Abbott ya hace cerca de treinta y tres años, en noviembre de 1983, Abbott publica su *artículo Program Design by Informal English Descriptions* [49] en este artículo se presenta un enfoque muy interesante sobre el diseño de programas orientados a objetos, escritos en ADA a partir de descripciones informales del inglés.

La principal contribución de Abbott es la relación propuesta entre los nombres y los tipos de datos. Esta relación permite reconocer a partir de la descripción de una solución en términos del dominio, los objetos y las operaciones sobre estos objetos que debe estar en capacidad de soportar una aplicación.

El enfoque de Abbott consiste de tres etapas. La primera es desarrollar una estrategia informal para el problema, la segunda es formalizar esa estrategia y finalmente la tercera consiste en segregar la solución en dos partes; un paquete y un subprograma para una colección de subprogramas. El desarrollo de una estrategia informal se fundamenta en expresar la solución del problema en el mismo nivel conceptual del problema mismo, esto es expresarse en lenguaje natural y lenguaje del dominio. En cuanto a formalizar la estrategia informal, esto es formalizar la solución por medio de la formalización de sus tipos de datos, objetos, operaciones y estructuras de control. La última consiste en dividir la

solución en dos partes fundamentales, un paquete y un subprograma para una colección de subprogramas; el paquete contiene la formalización del problema del dominio, es decir los tipos de datos y sus operadores; los subprogramas contienen los pasos específicos para resolver el problema en particular. En la figura 19 se puede observar más claramente estas tres etapas.

Figura 19. Etapas del diseño de aplicaciones por descripciones informales del inglés de Abbott

La segunda etapa del enfoque de Abbott es de especial interés para la presente investigación, durante esta se intenta descubrir los objetos y operaciones que van a hacer parte del modelo del dominio de la aplicación. Este descubrimiento de objetos y operadores se hace por medio de la identificación de nombres y verbos dentro de la estrategia informal redactada en la primera etapa del proceso de la figura 19.

Abbott plantea descubrir los tipos de datos, operadores etc., partiendo de las descripciones informarles, hechas en inglés. En el caso planteado por esta investigación, ReFree parte de una aplicación terminada donde los tipos de datos, operadores y atributos ya están plenamente definidos, solo que se intentan descubrir a través de su interfaz gráfica de usuario, con el fin de poder construir una descripción informal de la solución, expresada en características.

Lo anterior se basa en la idea fundamental de que la mayoría de tipos más representativos para el modelo de negocio son aquellos que están siendo manipulados por los usuarios desde la GUI, además un buen número de las operaciones sobre estos tipos de datos significativos, se pueden realizar por medio de las acciones proporcionadas al usuario también a través de la GUI.

Otro importante argumento para considerar la posibilidad de extraer estos objetos y operaciones a través del estudio de la GUI, es que en la mayoría de los casos la GUI es diseñada y construida teniendo en cuenta el lenguaje de los usuarios y en especial el lenguaje del dominio de la aplicación. Entonces es factible usar dicho

lenguaje para encontrar nombres y verbos relevantes que indiquen objetos y operadores presentes dentro del sistema.

Teniendo estos nombres y verbos identificados y asociados a alguna acción dentro de la GUI y sabiendo que las características son descripciones informales del sistema en lenguaje natural, se puede utilizar este inventario de términos para construir estas descripciones y entonces documentar nuevamente los requerimientos que la aplicación cumple intrínsecamente pero que dejan de ser evidentes a la hora de realizar un nuevo desarrollo en base a un sistema heredado.

5.5.1 Tercera revisión Matriz de Componentes

En esta revisión se consideran dos aspectos importantes para una nueva versión de la matriz de componentes, la primera es el conjunto de los elementos del lenguaje que se va a buscar y la segunda es un elemento de diseño que puede tenerse en cuenta para aumentar la cantidad de información obtenida dentro del análisis.

5.5.1.1 Cambio de encabezados

Los encabezados de la Matriz de Componentes cambian en esta última versión para mostrar lo que se busca dentro de la interfaz, es decir el nombre u objeto sobre el cual se está realizando la acción y la acción en sí o verbo. Para las dos últimas columnas, se modifican los campos de Relación y Función, por Función y Observaciones. En el campo de Función se debe registrar una descripción simple de la acción que se sucede al momento de activar el control que se está registrando. Mientras que en el campo de Observaciones se anotan detalles importantes a tener en cuenta, por ejemplo si el control hace parte de un grupo de controles mayor, si este despliega una interfaz cual es esta interfaz, o cualquier tipo de información relevante para el analista.

5.5.1.2 Inclusión de los Patrones de Diseño de Interfaces Gráficas

En este aspecto y de acuerdo a lo referenciado en la sección 4.1.3.2 Modelos de Interacción, es deseable que dentro de la Matriz exista la posibilidad de registrar el tipo de modelo de interacción que este utiliza, lo que permite tener una idea general de las características que tiene la aplicación en determinadas vistas de la misma, las cuales están caracterizadas en la tabla 5. Características patrones de diseño GUI.

Por lo anterior la tercera y última versión de la Matriz de Componentes se presenta en la figura 20, a continuación. En ella se pueden observar los cambios descritos anteriormente, además de una región para agregar anotaciones al final de la Matriz de Componentes.

Figura 20. Tercera versión Matriz de Componentes

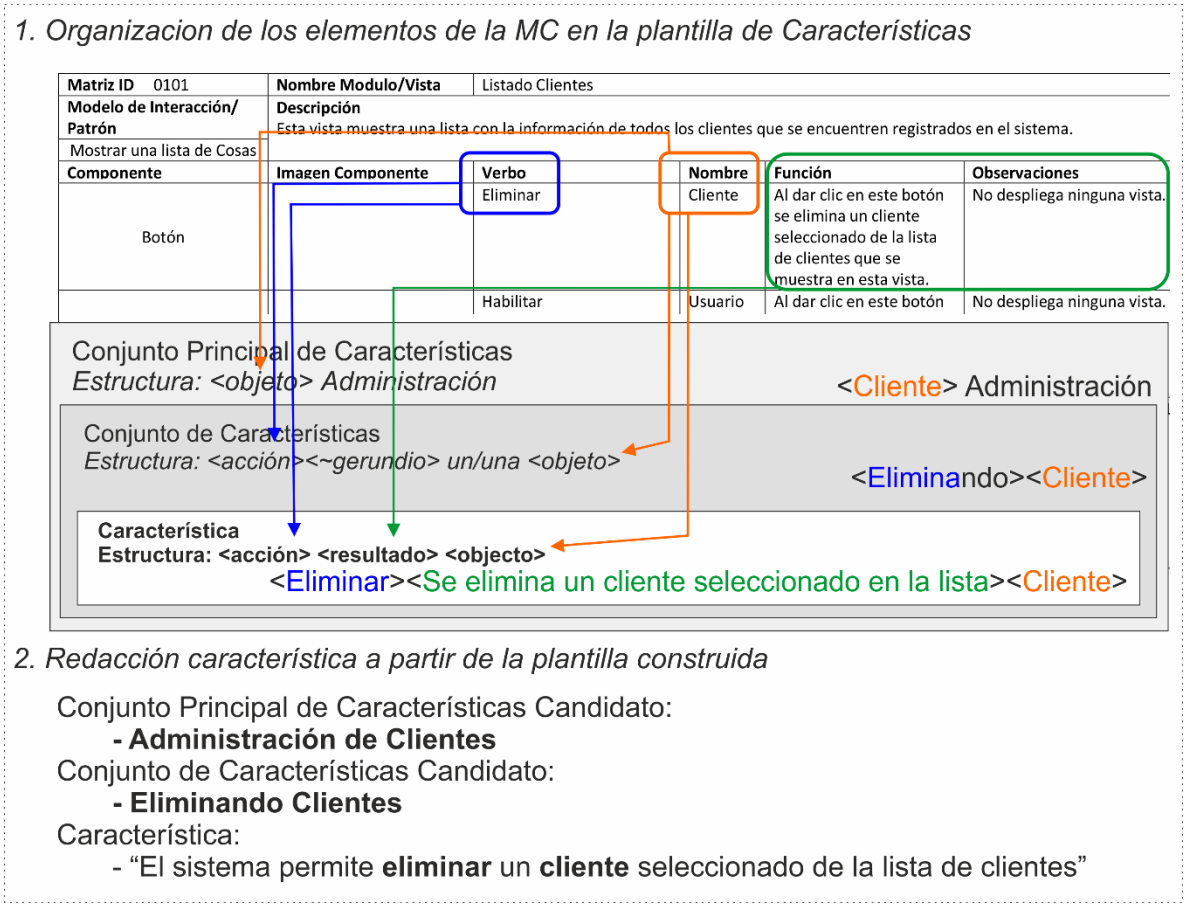
Matriz ID 0101	**Nombre Modulo/Vista**	Listado Clientes			
Modelo de Interacción/ Patrón	**Descripción** Esta vista muestra una lista con la información de todos los clientes que se encuentren registrados en el sistema.				
Mostrar una lista de Cosas					
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Botón		Eliminar	Cliente	Al dar clic en este botón se elimina un cliente seleccionado de la lista de clientes que se muestra en esta vista.	No despliega ninguna vista.
Botón		Habilitar	Usuario	Al dar clic en este botón se habilita un cliente seleccionado de la lista de clientes que se muestra en esta vista.	No despliega ninguna vista.
Botón		Crear	Usuario	Al dar clic sobre este botón se despliega inmediatamente una nueva vista, la cual contiene un formulario con todos los datos requeridos para crear un nuevo cliente.	ID Vista desplegada: 01010 Nombre Vista: Nuevo Usuario

5.5.2 Traducción a Características desde la Matriz de Componente

Esta parte de la cuarta iteración, se logra gracias al estudio de enfoques como el de Abbott en cuando al uso del lenguaje en el diseño de aplicaciones y al de las características como técnica de requerimientos, se propone entonces la construcción de las características usando dos elementos: Primero; los verbos y nombres hallados en el análisis de interfaces y segundo; la plantilla para construir los listados de características provista por la FDD y expuesta en la sección 4.1.4.1 Construcción Lista de Características. ReFree toma los verbos, nombres y descripciones de las funciones y los utiliza como elementos de construcción en la plantilla de características. En la figura 21 se puede observar el proceso de cómo, tomando los elementos destacados (verbo, nombre, función) y asignándolos dentro de las estructuras propuestas, se logran construir los conjuntos de características, sus nombres, y las características en sí.

Dentro de la matriz de componentes cada uno de los registros realizados genera una característica, la agrupación de las mismas se hace teniendo en cuenta el criterio del analista apoyado en los conjuntos principales de características y los conjuntos de características generados.

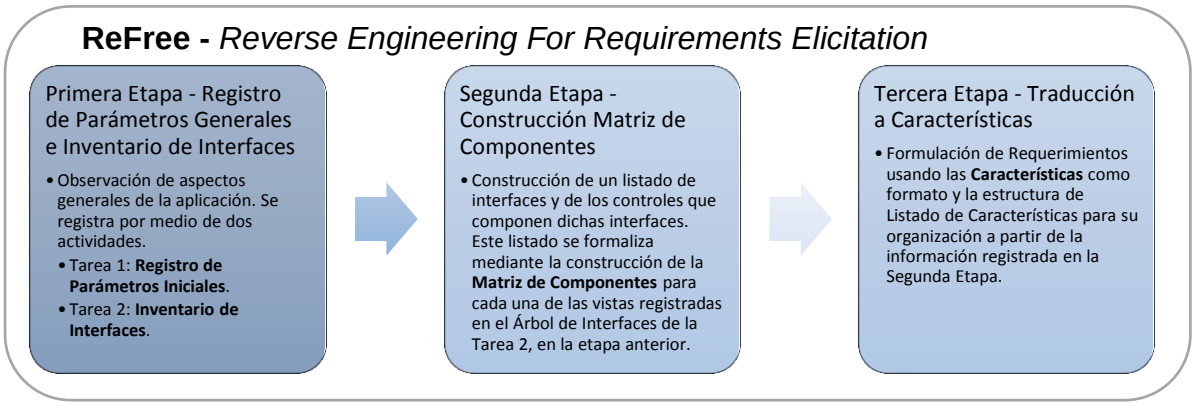
Figura 21. Construcción de una Características



A medida que se avanza en cada uno de los componentes que forman las interfaces de la aplicación heredada, se va construyendo de manera sistemática una lista de características, la cual es una descripción en lenguaje natural de las diferentes funcionalidades con las que cuenta la aplicación analizada.

Finalmente reorganizando se tiene el diagrama de bloques que presenta de manera general la técnica ReFree y los distintos aspectos que la conforman. Ver figura 22. En este se pueden observar claramente las tres etapas que conforman ReFree y de manera general las actividades que se desarrollan dentro de cada etapa.

Figura 22. Diagrama de bloques final ReFree



6. DESCRIPCIÓN Y APLICACIÓN DE LA TECNICA REFREE

En la primera parte de este capítulo se presenta una descripción de la técnica y el conjunto de heurísticas que se desarrollaron para las principales actividades del método propuesto, ReFree. En las secciones subsiguientes se exponen los ejemplos del uso de la técnica en aplicaciones con diversas funcionalidades y dominios de aplicación.

6.1 DESCRIPCIÓN DE LA TÉCNICA REFREE

A continuación se presenta un paso a paso de ReFree, en esta sección se explica de manera sencilla cada una de las actividades que debe seguir el analista para obtener a través de la aplicación de ReFree, los requerimientos de la aplicación heredada en el formato de características, el diagrama de bloques general de la técnica se puede ver en la figura 24.

Como consideraciones preliminares, el analista debe tener acceso a la aplicación con una cuenta que le permita acceder plenamente a los módulos sobre los cuales va a aplicar ReFree. En el caso de aplicaciones con diferentes niveles de acceso, este debe contar con el mayor porcentaje de acceso posible sin ir en contra de las políticas de seguridad de la empresa, sin embargo es responsabilidad del cliente poner a disposición los recursos necesarios para realizar cualquier proceso de captura de requerimientos.

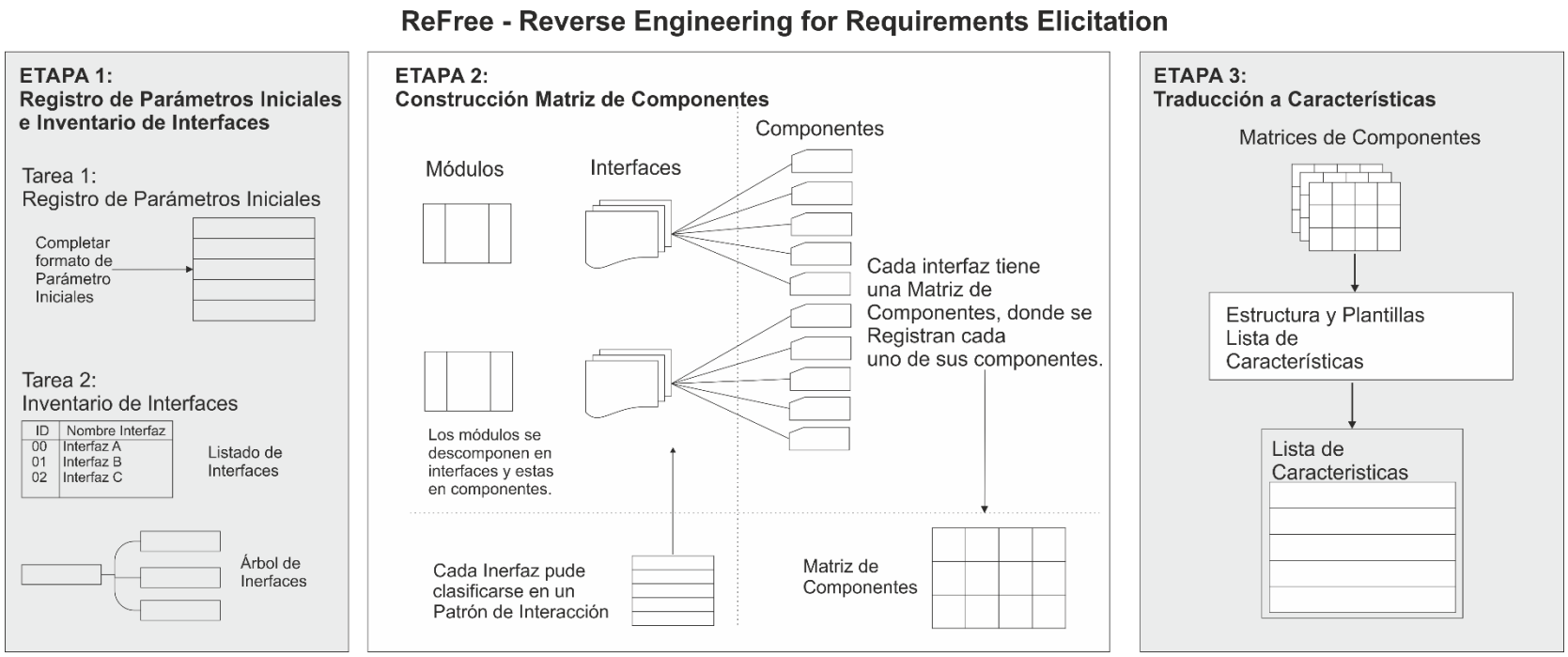
6.1.1 Primera Etapa: Parámetros generales e inventario de interfaces

En esta primera etapa deben desarrollarse dos tareas: el Registro de Parámetros Iniciales y el Inventario de Interfaces.

Figura 23. Heurísticas primera etapa, tarea uno ReFree

- El registro de parámetros iniciales es una forma ordenada y simple de registrar lo que parece obvio. Estos datos preliminares dan una idea al analista del dominio en el que se adentra a través de la interacción con esta aplicación.
- La información obtenida en esta primera etapa proviene de la interacción con algún usuario o encargado, y en la mayoría de los casos es la única información que se obtiene a través de entrevistas breves.

Figura 24. Diagrama de Bloques ReFree



6.1.1.1 Tarea Uno: Registro de parámetros iniciales

Realizando una observación inicial de la aplicación se debe registrar la información requerida en el Formato de Parámetros Iniciales. En el Anexo C se encuentra una versión imprimible de este formato.

6.1.1.2 Tarea Dos: Inventario de interfaces

El inventario de interfaces consiste en la recopilación y organización de las interfaces que van a ser sometidas a análisis con ReFree. En esta tarea se asignan los identificadores a cada una de las interfaces, se sugiere tener en cuenta la plantilla de modelos de navegación Figura 26. Esta guía muestra los modelos de navegación más comunes en los sistemas software y las diferentes maneras para asignar los identificadores únicos de las interfaces que conforman la aplicación. Versión para imprimir Anexo C. El uso de estos modelos e identificadores no es obligatorio, el analista puede usarlos o adaptarlos de acuerdo a sus necesidades.

Al asignar los identificadores a las interfaces también se construye un listado de todas las interfaces que van a ser sometidas a análisis, este se desarrolla siguiendo la forma en la que se navega dentro de la aplicación, esto quiere decir teniendo en cuenta los niveles de profundidad del sistema, y registrando el identificador y nombre de cada una de las vistas. Al observar este listado debe ser posible reconocer el camino que se sigue para llegar a una determinada vista de la interfaz.

Para facilitar mucho más la visualización de este listado, ReFree plantea la construcción de un árbol de interfaces, el cual es simplemente una representación gráfica de la lista de interfaces. El listado inicial de interfaces se recomienda sea realizado de manera plana en cualquier hoja de cálculo y luego ser estructurado en una herramienta para la construcción de diagramas (árbol de interfaces), para efectos de los ejemplos presentados en este documento las herramientas usadas fueron Microsoft Excel y Microsoft Visio.

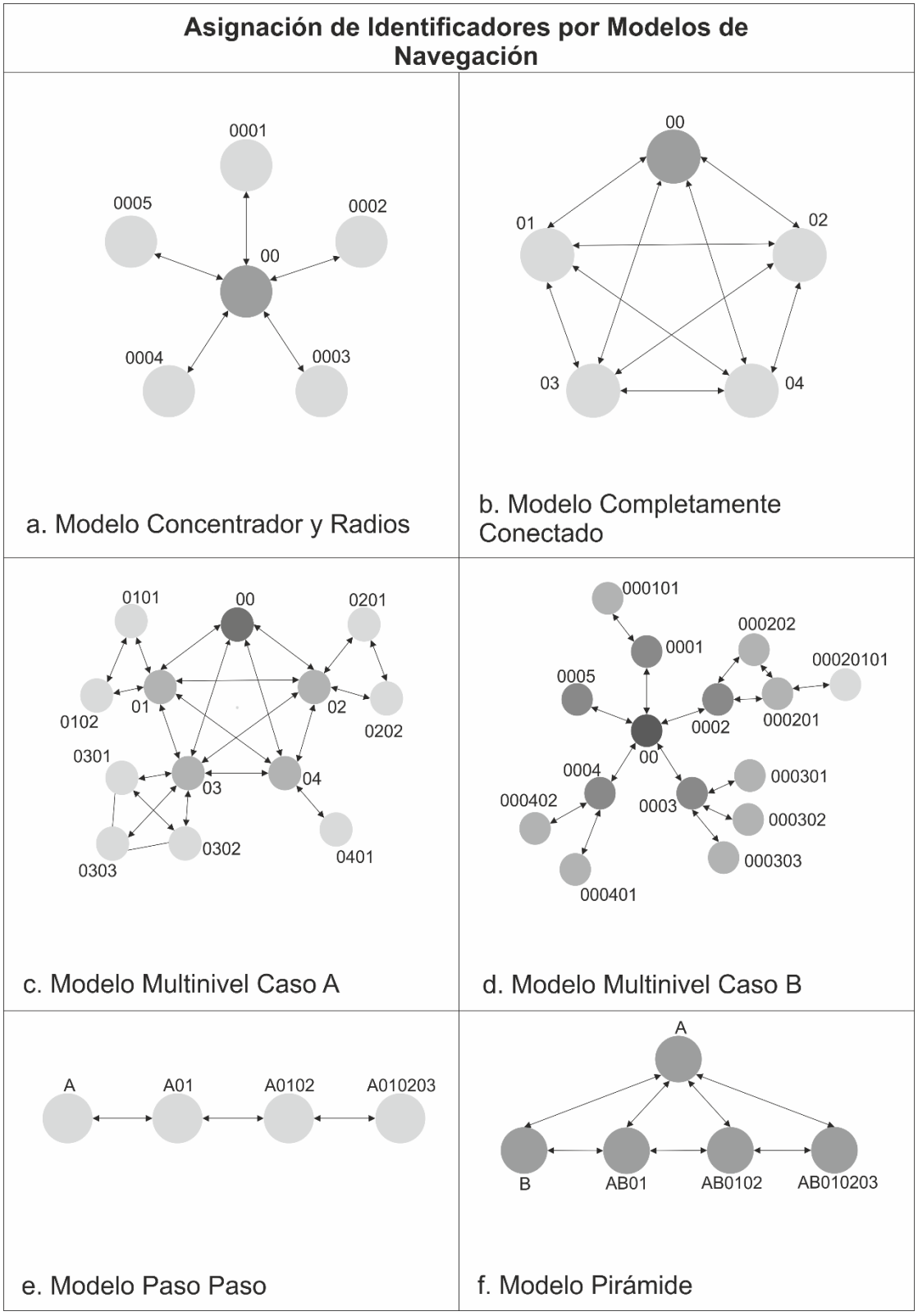
Figura 25. Heurísticas primera etapa, tarea dos ReFree

- El listado de interfaces permite no solo registrar los identificadores y nombres de las interfaces, también permite a través de otras herramientas simples como su estructura, los colores y hasta el tipo de letra usado, registrar eventos importantes y repetitivos dentro del sistema. Errores, falta de datos, uso y hasta diseños repetitivos son reconocibles y documentados de manera práctica y sencilla.

- El árbol de interfaces que se deriva de la lista de interfaces es un representación visual de la misma y permite al analista comprender de una manera esquemática los distintos lugares que va a recorrer en su análisis y como se vinculan entre ellos.

- Es importante el uso de herramientas sencillas en esta etapa, no se debe incurrir en costosas aplicaciones ni en tiempo para el aprendizaje de las mismas, la idea es que sea tan natural como usar lápiz y papel para realizar anotaciones sobre el sistema.

Figura 26. Guía modelos de navegación



6.1.2 Segunda etapa: Construcción matriz de componentes

Para cada una de las interfaces incluidas dentro del inventario de interfaces del paso anterior, se debe hacer un registro detallado de todos los elementos que la componen. Para esto es necesario completar la Matriz de Componentes, el formato de dicha matriz se encuentra disponible para impresión en el Anexo C. A continuación se muestra una imagen de este formato, figura 27.

Figura 27. Formato matriz de componentes

Matriz ID	Nombre Modulo/Vista				
Modelo de Interacción/ Patrón	Descripción				
Componente	Imagen Componente	Verbo	Nombre	Función	Observaciones

Como cada una de las interfaces que se registra en el paso anterior debe contar con una Matriz de Componentes, en el campo **Matriz ID** ubicado en la parte superior izquierda del formato, debe registrarse el Identificador de la interfaz que fue asignado en la etapa anterior a cada una de las interfaces del sistema. El campo **Nombre Módulo/Vista** debe contener el nombre del módulo y /o de la vista a la cual pertenece dicha Matriz de Componentes, el cual también debe coincidir con el que se encuentra registrado en el Árbol del Interfaces.

Para el campo **Modelo de Interacción/Patrón** se debe tener en cuenta la plantilla de los modelos de interacción suministrada por ReFree, ver Anexo C. En este campo se consigna el nombre del patrón de modelo de interacción que tiene la interfaz y a continuación en el campo llamado **Descripción** se debe anotar la descripción del patrón dada por la plantilla e información extra sobre la interfaz, que permita reconocer su funcionamiento general. De no haber una coincidencia clara con alguno de los patrones incluidos el analista está en plena libertad de realizar las notas descriptivas que crea necesarias.

En cuanto a las columnas de la matriz presentes, en el formato de izquierda a derecha se encuentra primero la columna **Componente** en la cual se ingresa el nombre del tipo de componente que se va a registrar, sea un botón, caja de texto, etiquetas, etc. La segunda columna I**magen Componente** debe contener como su nombre lo dice la imagen del componente en cuestión, la tercera columna Verbo almacena el verbo que describe la acción que realiza el sistema al accionarse el componente de la interfaz que se está registrando.

En la columna **Nombre** se consigna el nombre del objeto sobre el cual se está realizando la acción, la siguiente columna **Función** debe contener una descripción breve y clara de la acción que se sucede al momento de la interacción con el control. La última columna **Observaciones** es usada fundamentalmente para anotar, si es el caso, qué vista se despliega al accionar el control, para eso se incluyen en este campo el ID de la vista desplegada y el nombre de la vista, este espacio también puede ser usado por el analista para realizar cualquier observación aclaratoria sobre el componente que se está registrando.

Figura 28. Heurísticas segunda etapa ReFree

- La matriz de componentes es una manera ordenada y sistemática de registrar lo observable en la interfaz gráfica de una aplicación.
- Los patrones de interacción permiten clasificar y describir de lo que de manera general sucede en una determinada interfaz.
- En la interfaz gráfica es posible reconocer palabras clave del lenguaje del dominio de la aplicación y por lo tanto del lenguaje usado por los usuarios.
- Se reconoce a los **Nombres** encontrados dentro de la interfaz como **Objetos** candidatos del modelo del dominio de la aplicación.
- Se reconoce a las **Funciones** que se suceden dentro de la interfaz como operaciones que se realizan sobre los **Objetos.**
- Se reconoce a los datos manipulados por el sistema a través de formularios como atributos de los objetos del sistema.

Terminada la construcción de la Matriz de Componentes para todas y cada una de las interfaces incluidas en el inventario de interfaces se procede a la última y tercera etapa de la técnica ReFree la cual consiste en la construcción del listado de características.

6.1.3 Tercera etapa: Traducción a características

Después de completar el registro de los componentes de la interfaz gráfica se procede a realizar la traducción de cada uno de los registros que componen la Matriz de Componentes en características del sistema, las cuales como ya se ha establecido anteriormente representan requerimientos de la aplicación.

La entrada más importante de este proceso son las distintas matrices de componentes construidas en la etapa anterior. Para cada una de estas matrices se debe obtener una pequeña lista de características, la cual describe las diferentes

funciones que el sistema lleva a cabo a través de dicha interfaz y su interacción con el usuario.

Esta etapa consiste simplemente en utilizar la plantilla existente para la construcción de características utilizando los elementos recolectados y organizados en las etapas anteriores de ReFree, de manera más precisa los elementos contenidos en las matrices de componentes.

En la figura 29 se puede observar en colores, como los distintos elementos registrados en la matriz de componentes son utilizados en la plantilla para generar características, conjuntos de características y conjuntos mayores de características. Todos estos conjuntos y características son inicialmente "candidatos", esto quiere decir que debe realizarse una revisión para depurar los resultados obtenidos. Para esto y con ayuda del árbol de interfaces y la pericia de analista, se recomienda re-organizar las características obtenidas en los conjuntos y conjuntos mayores de características que sean seleccionados definitivamente para representar la jerarquía de requerimientos obtenida.

Figura 29. Plantilla traducción a características

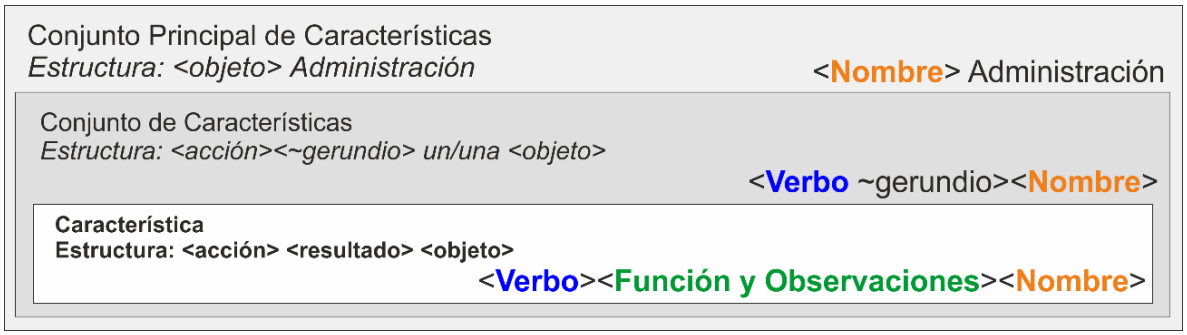


Figura 30. Heurísticas tercera etapa ReFree

- Los Verbos y Nombres extraídos por medio de la observación y registro en la matriz de componentes se convierten en elementos estructurales claves a la hora de redactar las descripciones en forma de características del sistema.
- Nombres y Verbos distribuidos en la interfaz gráfica son extraídos y convertidos en frases naturales de entendimiento para todos los interesados.
- Por medio de la descomposición simple de la interfaz y el reconocimiento de elementos del lenguaje que maneja, es posible tener una base para obtener otros productos como diagramas de clase, casos de uso, etc.

6.2 EJEMPLOS DE APLICACIÓN

En esta sección se realiza una presentación de algunas pruebas realizadas con ReFree, las aplicaciones son seleccionadas simplemente teniendo en cuenta que estas pertenezcan a dominios de aplicación muy diferentes. No se consideran plataformas, tecnologías de desarrollo, licencias, etc.

Las pruebas se realizan de manera parcial sobre alguna de las interfaces de cada una de las aplicaciones.

6.2.1 VTiger

Esta aplicación es una de las primeras en analizarse incluso es sometida a análisis con las primeras versiones de ReFree, a continuación se presenta el estudio hecho con la versión final de la técnica para el módulo Calendario de la aplicación.

Primera Etapa: Registro de Parámetros Generales e Inventario de Interfaces

Tarea uno: Registro de Parámetros Iniciales

Figura 31. Formato de parámetros iniciales VTiger

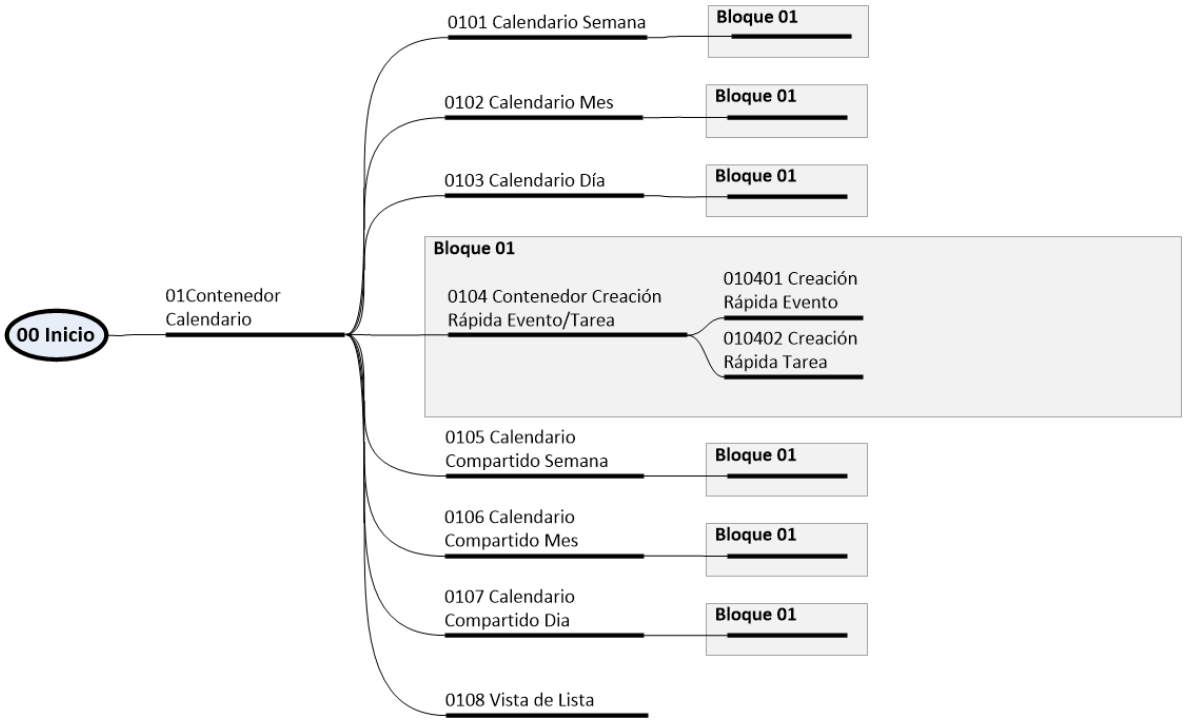
Formato Parámetros Iniciales	
Nombre de la aplicación: VTiger	**Fecha:** 05/04/2015
Tipo de aplicación: Web	
S.O sobre el cual se ejecuta: Windows 8 en el navegador Google Chrome	
Propósito u objetivo de la aplicación: La aplicación es una CRM (*Client Resource Management*) de código abierto, es decir un software para la administración de la relación con los clientes, su objetivo principal es gestionar las relaciones con los clientes y las actividades que conlleven esta relación, como por ejemplo órdenes de compra, facturas, productos, oportunidades, presupuesto, etc. .	
Quienes son los usuarios de la aplicación: Empleados, empresas interesadas en el manejo asistido de la relación con el cliente. Sin embargo para este caso el uso es dado únicamente por el desarrollador de este proyecto.	
Forma en que se usa la aplicación: En este momento se está usando como una de las plataformas de prueba para la técnica ReFree.	
Dónde se usa la aplicación: Uso en un único computador personal.	
Número de Módulos para analizar: Uno (1) – Modulo Calendario	**Número de Formatos para analizar:** Once (11)
Observaciones: Para efectos del desarrollo del ejemplo se tienen en cuenta solo algunas interfaces, estas están determinadas por el árbol de interfaces que se desarrolla más adelante.	

Tarea Dos: Inventario de Interfaces

Tabla 10. Listado de interfaces VTiger

Módulo Calendario Vtiger – Listado de Interfaces				
00 Inicio				
01 Contenedor Calendario	0101 Calendario Semana	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0102 Calendario Mes	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0103 Calendario Día	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0104 Contenedor Creación Rápida Evento/Tarea	010401 Creación Rápida Evento		
		010402 Creación Rápida Tarea		
	0105 Calendario Compartido Semana	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0106 Calendario Compartido Mes	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0107 Calendario Compartido Día	*0104 Contenedor Creación Rápida Evento/Tarea*	*010401 Creación Rápida Evento*	*01040101 Creando Nuevo Evento*
			010402 Creación Rápida Tarea	*01040102 Creando Nuevo Calendario*
	0108 Vista de Lista			

Figura 32. Árbol de interfaces VTiger



Segunda etapa: Construcción Matriz de Componentes

Para cada una de las vistas identificadas en el inventario de interfaces se construye una Matriz de Componentes. En este caso debido a la similitud entre las interfaces este trabajo es resumido en sólo seis matrices de componentes. En cada una de ellas se explican las vistas con similitudes y el manejo de cada una de estas.

Tabla 11. Matriz de componentes 01 VTiger

Matriz ID 01	**Nombre Modulo/Vista**	Contenedor Calendario			
Modelo de Interacción/ Patrón Vistas Alternativas	**Descripción** La aplicación permite que el usuario escoja entre vistas alternativas que son substancialmente diferentes de la vista por defecto. La interfaz cuenta con un "interruptor" para cambiar el modo de visualización. En estas vistas alternativas alguna información puede ser añadida y alguna puede ser retirada, sin embargo el contenido básico sigue siendo el mismo. En este caso el contenedor presenta vistas alternativas del Calendario y Calendario Compartido, los calendarios se usan para el manejo de una agenda del usuario y otra compartida con otros usuarios.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**

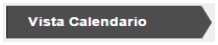
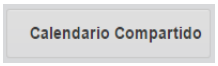
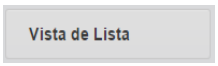
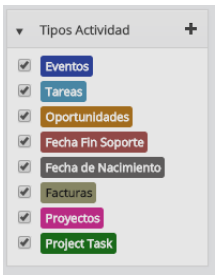
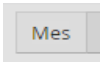
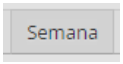
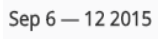
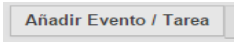
Button		Mostrar	Calendario	Muestra dentro de la interfaz 01 la vista el calendario.	Vista desplegada y por defecto: 0101
Button		Mostrar	Calendario Compartido	Muestra dentro de la presente interfaz (01) la vista de calendario compartido semanal	Vista desplegada: 0105
Button		Mostrar	Listado Actividades	Muestra dentro de la presente interfaz (01) el listado de Eventos/Tareas del usuario.	Vista Desplegada: 0108
Collapsible Panel		Mostrar	Detallado de las actividades registradas o modificadas recientemente en el calendario.	El panel muestra una lista de los nombres de las actividades al dar clic en los nombres se despliega una vista con la información detallada de la actividad.	Este es un acceso rápido a las últimas actividades agregadas o modificadas.
Collapsible Panel		Mostrar/Ocultar	Tipos de Actividad	El panel contiene un grupo de Check Box para seleccionar los tipos de actividad que quieren ser vistos en el calendario.	
Notas:					

Tabla 12. Matriz de componentes 0101 VTiger

Matriz ID 0101	**Nombre Modulo/Vista**	Calendario Semana			
Modelo de Interacción/ Patrón Vistas Alternativas	**Descripción** La aplicación permite que el usuario escoja entre vistas alternativas que son substancialmente diferentes de la vista por defecto. La interfaz cuenta con un "interruptor" para cambiar el modo de visualización. En estas vistas alternativas alguna información puede ser añadida y alguna puede ser retirada, sin embargo el contenido básico sigue siendo el mismo. En este caso el contenedor presenta vistas alternativas del Calendario, este se presenta semanal, mensual o diario.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Button		Cambiar	Vista	Al presionar el botón el calendario pasa de su vista semanal a vista por mes.	Vista desplegada: 0102
Button		Cambiar	Vista	Al presionar el botón el calendario pasa de su vista semanal a vista por semana (vista por defecto).	Vista desplegada: 0101
Button		Cambiar	Vista	Al presionar el botón el calendario pasa de su vista semanal a vista por día.	Vista Desplegada: 0103
Label		Mostrar	Intervalo de fechas	Muestra el intervalo de fechas para la semana que se está observando en el calendario.	
Button		Reubicar	La fecha actual	Reubica al usuario en vista del calendario donde se encuentra la fecha actual cuando este se desplaza a otras fechas en el calendario.	
Button		Mostrar	Vista	Muestra como diálogo modal la vista Contenedor Creación Rápida Evento/Tarea, desplegando por defecto la vista Creación Rápida Evento.	Vista Desplegada: 0104, por defecto esta contiene 010401

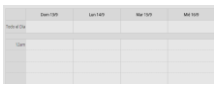
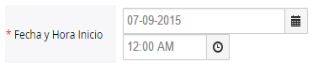
Button		Desplazar	Calendario	Desplaza la vista de calendario horizontalmente.	
Button		Mostrar	Vista	Muestra como diálogo modal la vista Configurar Calendario.	
Schedule		Mostrar	Actividades	Muestra las actividades registradas en el calendario, y permite realizar acciones (agregar, eliminar, editar) sobre las mismas. Al hacer clic sobre los campos de hora y día muestra como diálogo modal la vista Contenedor Creación Rápida Evento/Tarea, desplegando por defecto la vista Creación Rápida Evento.	Vista Desplegada: 0104, por defecto esta contiene 010401
Nota: Las Matrices de Componentes pertenecientes a las interfaces 0102, 0103, 0105, 0106 y 0107 son esencialmente la misma, las variaciones consisten en la forma que se despliega la información y el tipo de información. Para las interfaces 0102 y 0103 el componente *Schedule* se muestra configurado para una vista del mes o del día respectivamente. Las interfaces 0105, 0106, 0107 tienen también configuraciones de semana, mes y día. Estas muestran las actividades compartidas con otros usuarios del sistema. Por lo anterior se omite el desarrollo de las matrices de componentes para estas interfaces.					

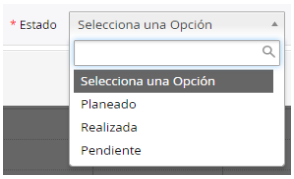
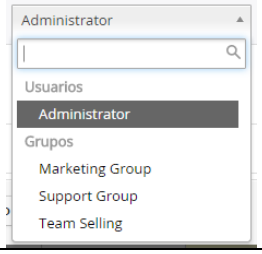
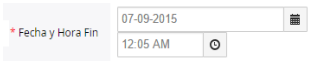
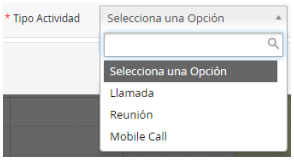
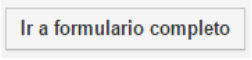
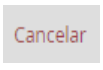
Tabla 13. Matriz de componentes 0104 VTiger

Matriz ID 0104	**Nombre Modulo/Vista**	Contenedor Creación Rápida Evento/Tarea			
Modelo de Interacción/ Patrón Múltiples espacios de trabajo	**Descripción** La aplicación permite desarrollar múltiples tareas al tiempo, el usuario puede ir de un proceso a otro sin tener que finalizar alguno de los procesos, este patrón se distingue por la presencia de pestañas (tabs) en el nivel superior, grupos de pestañas, múltiples ventanas donde los usuarios pueden ver diferentes documentos, proyectos, archivos o contextos al mismo tiempo. (Pestañas, ventanas de sistema operativo separadas, columnas o paneles dentro de una ventana, ventanas divididas. En este caso se cuenta con dos botones que permiten cargar dos vistas con funciones diferentes,				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Button		Seleccionar	Vista Evento	Carga dentro de la vista la interfaz 010401, formulario para registrar de manera rápida la	Vista desplegada y por defecto: 010401 Este control no contiene en su etiqueta el verbo de

				información de un nuevo evento en el calendario.	la acción sino el Nombre del objeto a manipular.
Button		Seleccionar	Vista Tarea	Carga dentro de la vista la interfaz 010402, formulario para registrar de manera rápida la información de una nueva tarea en el calendario.	Vista desplegada: 010402 Este control no contiene en su etiqueta el verbo de la acción sino el Nombre del objeto a manipular.
Nota:					

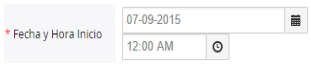
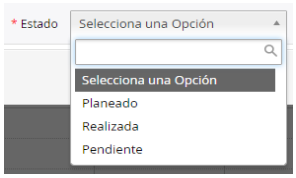
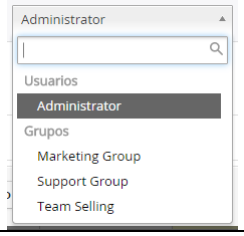
Tabla 14. Matriz de componentes 0104001 VTiger

Matriz ID 010401	**Nombre Modulo/Vista**	Creación Rápida de Evento			
Modelo de Interacción/ Patrón Múltiples espacios de trabajo	**Descripción** La aplicación permite desarrollar múltiples tareas al tiempo, el usuario puede ir de un proceso a otro sin tener que finalizar alguno de los procesos, este patrón se distingue por la presencia de pestañas (tabs) en el nivel superior, grupos de pestañas, múltiples ventanas donde los usuarios pueden ver diferentes documentos, proyectos, archivos o contextos al mismo tiempo. (Pestañas, ventanas de sistema operativo separadas, columnas o paneles dentro de una ventana, ventanas divididas. En este caso se cuenta con dos botones que permiten cargar dos vistas con funciones diferentes.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Text Box		Ingresar	Asunto	Espacio para ingresar el nombre del asunto del nuevo evento a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Date Time Picker		Seleccionar	Fecha y Hora	Estos controles permiten seleccionar tanto la fecha y la hora para el inicio del nuevo evento a registrar.	Este control permite ingresar los atributos del elemento manipulado.

Combo Box		Seleccionar	Estado	Seleccionar el estado para el nuevo evento a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Combo Box		Seleccionar	Quien	Seleccionar a quien está asignada la tarea.	Este control permite ingresar los atributos del elemento manipulado.
Date Time Picker		Seleccionar	Fecha y Hora	Estos controles permiten seleccionar tanto la fecha y la hora para el fin del nuevo evento a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Combo Box		Seleccionar	Tipo de Actividad	Selecciona el tipo de evento que se está creando en el sistema.	Este control permite ingresar los atributos del elemento manipulado.
Button		Mostrar	Vista Creación de Evento	Muestra una vista completa para la creación de un nuevo evento.	
Button		Guardar	Información	Se registra la información del nuevo evento en el sistema.	
Button		Cancelar	Creación Nuevo Evento	Sale de la vista y no registra información.	
Nota: Todos los campos requeridos en esta vista son obligatorios, si no están todos completos, el sistema no permite registrar (guardar) el nuevo evento. Estos son los atributos mínimos que deben estar asignados					

al evento. Si el usuario desea puede pasar de esta vista a una con más completa, donde puede registrar más atributos al evento.

Tabla 15. Matriz de componentes 010402 VTiger

Matriz ID 010402	**Nombre Modulo/Vista**	Creación Rápida Calendario (el sistema parece mostrar erróneamente la palabra calendario en vez de Tarea)			
Modelo de Interacción/ Patrón Múltiples espacios de trabajo	**Descripción** La aplicación permite desarrollar múltiples tareas al tiempo, el usuario puede ir de un proceso a otro sin tener que finalizar alguno de los procesos, este patrón se distingue por la presencia de pestañas (tabs) en el nivel superior, grupos de pestañas, múltiples ventanas donde los usuarios pueden ver diferentes documentos, proyectos, archivos o contextos al mismo tiempo. (Pestañas, ventanas de sistema operativo separadas, columnas o paneles dentro de una ventana, ventanas divididas. En este caso se cuenta con dos botones que permiten cargar dos vistas con funciones diferentes.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Text Box		Ingresar	Asunto	Espacio para ingresar el nombre del asunto de la nueva tarea a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Date Time Picker		Seleccionar	Fecha y Hora	Estos controles permiten seleccionar tanto la fecha y la hora para el inicio de la nueva tarea a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Combo Box		Seleccionar	Estado	Seleccionar el estado para la nueva tarea a registrar.	Este control permite ingresar los atributos del elemento manipulado.
Combo Box		Seleccionar	Quien	Seleccionar a quien está asignada la tarea.	Este control permite ingresar los atributos del elemento manipulado.
Date Picker		Seleccionar	Fecha Vencimient	Seleccionar la fecha de	Este control permite

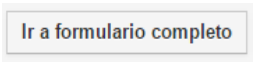
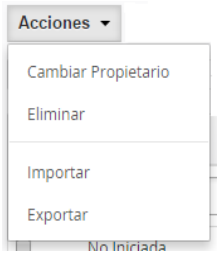
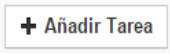
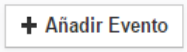
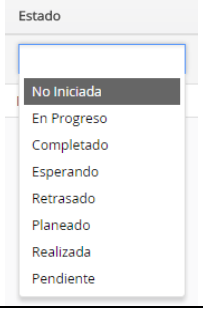
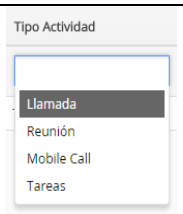
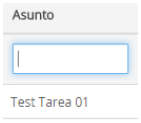
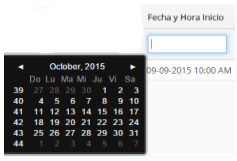
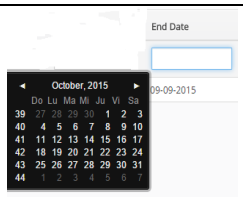
			o	vencimiento de la tarea.	ingresar los atributos del elemento manipulado.
Button		Mostrar	Vista Creación de Evento	Muestra una vista completa para la creación de un nuevo evento.	
Button		Guardar	Información	Se registra la información del nuevo evento en el sistema.	
Button		Cancelar	Creación Nuevo Evento	Sale de la vista y no registra información .	
Nota: Todos los campos requeridos en esta vista son obligatorios, si no están todos completos, el sistema no permite registrar (guardar) el nuevo evento. Estos son los atributos mínimos que deben estar asignados al evento. Si el usuario desea puede pasar de esta vista a una con más completa, donde puede registrar más atributos al evento.					

Tabla 16. Matriz de componentes 0108 VTiger

Matriz ID 0108	**Nombre Modulo/Vista**	Vista de Lista			
Modelo de Interacción/ Patrón Destacado, Búsqueda y Navegación	**Descripción** El sitio o aplicación contiene tres elementos obligatoriamente, un elemento destacado, una caja de búsqueda y una lista de elementos o categorías navegables. La aplicación posee una lista extensa de elementos que interesan al usuario, debe facilitar su búsqueda, a su vez alguno de estos elementos debe ser destacado ya que puede ser de interés para el usuario. En esta vista se presenta una lista de las actividades registradas en el calendario, y diferentes herramientas así como criterios de búsqueda.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Combo Box		Seleccionar	Acciones	Despliega un menú conjunto de acciones disponibles para realizar sobre las actividades mostradas en la lista, que hayan sido seleccionadas.	Las tareas mostradas en la lista están acompañadas de un *Check Box* para ser seleccionadas.

Button		Mostrar	Vista	Muestra la interfaz para ingresar la información relacionada a la nueva tarea que se vaya a ingresar al sistema.	Vista desplegada: 01040201
Button		Mostrar	Vista	Muestra la interfaz para ingresar la información relacionada al nuevo evento que se vaya a ingresar al sistema.	Vista desplegada: 01040101
Button		Buscar	Actividad	Muestra las actividades registradas en el sistema, agrupadas alfabéticamente por el asunto que se le ha asignado a la actividad (Nombre dado a la actividad).	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda, la letra seleccionada por el usuario.
Button		Desplazar	Página	Configuración de tres botones que permite saltar a la página anterior, siguiente o la determinada por el usuario, esto en el caso que la lista de actividades ya tenga una extensión considerable.	
Combo Box		Filtrar	Actividad	Filtrar la búsqueda de actividades por el estado.	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda.

Combo Box		Filtrar	Actividad	Filtrar actividades por el tipo.	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda.
Combo Box		Filtrar	Actividad	Filtrar actividades por el asunto. (Nombre dado a la actividad).	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda.
Combo Box		Filtrar	Actividad	Filtrar actividad por relación.	El sistema en este momento no cuenta con la información necesaria para mostrar las opciones de búsqueda en este campo.
Date Picker		Filtrar	Actividad	Filtrar actividades por la fecha y hora de inicio.	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda.
Date Picker		Filtrar	Actividad	Filtrar actividades por la fecha de finalización.	Las actividades mostradas en la lista son las que cumplen con el criterio de búsqueda.
Nota:					

Tercera etapa: Traducción a Características

Utilizando la plantilla proporcionada para generar características, figura 29, se puede iniciar con la construcción y redacción de las características. Para cada una de las matrices de componentes se obtiene un conjunto principal y un conjunto de características candidatas, que al final del análisis deben ser evaluados y reclasificados Otros elementos importantes en la captura de requerimientos del sistema y que se deben tener muy en cuenta son los patrones de interacción identificados y las notas hechas por el analista para cada una de las interfaces incluidas en el estudio.

Matriz de Componentes 01
Nombre del Módulo o Vista: *Contenedor Calendario*
Conjunto Principal de Características Candidato: *Administración Calendario*
Conjunto de Características Candidato: *No Definido*

1. *La vista cuenta con botones que permiten cargar dentro de ella diferentes vistas del calendario personal y calendario compartido.*
2. *El sistema muestra la vista de calendario.*
3. *El sistema muestra la vista de calendario compartido.*
4. *El sistema muestra el calendario en forma de lista.*
5. *El sistema cuenta con un acceso rápido para mostrar la información detallada de las últimas actividades registradas o modificadas.*
6. *El sistema permite seleccionar que actividades mostrar u ocultar en el calendario de acuerdo al tipo de actividad.*

Matriz de Componentes 0101
Nombre del Módulo o Vista: *Calendario Semanal*
Conjunto Principal de Características Candidato: *Administración Calendario Semanal*
Conjunto de Características Candidato: *No definido*

1. *El sistema permite cambiar la vista calendario a vista calendario mensual.*
2. *El sistema muestra la vista calendario semanal por defecto.*
3. *El sistema permite cambiar la vista del calendario a vista calendario diario.*
4. *El sistema muestra el rango de fechas que está desplegando en la vista, sea por semana, mensual o diario.*
5. *El sistema reubica rápidamente su vista en la fecha actual, cuando se está navegando a través del calendario, en cualquiera de sus vistas (mensual, semanal, diaria).*
6. *El sistema muestra una vista rápida para la creación de un Evento/Tarea, mostrando por defecto la vista para crear un evento.*
7. *El calendario se desplaza horizontalmente por semanas, meses o días.*
8. *El sistema cuenta con una opción para configurar el calendario.*
9. *El sistema muestra gráficamente el calendario en vista mensual, semanal y diaria, con las actividades registradas y permite realizar acciones de agregar, eliminar y editar sobre las mismas. Al realizar clic sobre los campos del calendario, se despliega la vista para la creación rápida de un Evento.*
10. *El calendario debe poder manejarse de manera compartida con otros usuarios del sistema y realizar todas las actividades del calendario personal.*

Matriz de Componentes 0104
Nombre del Módulo o Vista: *Contenedor creación rápida Evento/Tarea*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *La vista contiene dos botones que permiten cargar dentro de ella dos formularios para crear de manera eventos o tareas en el sistema.*
2. *El botón Evento carga el formulario de registro de un evento, este formulario se carga por defecto siempre que se muestra esta vista.*
3. *El botón Tarea carga el formato de registro rápido de tarea.*

Matriz de Componentes 010401
Nombre del Módulo o Vista: *Creación Rápida de Evento*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *El sistema cuenta con un formulario para el registro rápido de un evento en el sistema.*
2. *El sistema permite registrar el asunto del nuevo evento.*
3. *El sistema permite seleccionar la fecha y hora para el inicio del nuevo evento.*
4. *El sistema permite seleccionar el estado para el nuevo evento (Planeado, Realizada, Pendiente).*
5. *El sistema permite seleccionar a quien se asigna el nuevo evento.*
6. *El sistema permite seleccionar la fecha y hora de fin para el nuevo evento.*
7. *El sistema permite seleccionar el tipo de evento (llamada, reunión, llamada móvil).*
8. *El sistema muestra un formulario completo para la creación completo para la creación de un nuevo evento. Este formulario solicita más información más información sobre el evento que el formulario de Creación Rápida de Evento.*
9. *El sistema permite guardar el registro del nuevo evento.*
10. *El sistema permite cancelar la creación del nuevo evento.*
11. *Todos los campos solicitados en este formulario son obligatorios ya que son los datos mínimos requeridos por el sistema para generar un nuevo evento, por esto son incluidos en la vista Creación Rápida de Evento.*

Matriz de Componentes 010402
Nombre del Módulo o Vista: *Creación Rápida de Calendario (el sistema parece mostrar erróneamente la palabra calendario en vez de Tarea)*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *El sistema cuenta con un formulario para el registro rápido de una tarea en el sistema.*
2. *El sistema permite registrar el asunto de la nueva tarea.*
3. *El sistema permite seleccionar la fecha y hora para el inicio de la nueva tarea.*
4. *El sistema permite seleccionar el estado de la nueva tarea (No iniciada, En Progreso, Completando, Esperando, Retrasado, Planeado, Realizada, Pendiente).*
5. *El sistema permite seleccionar a quien está asignada la tarea.*

6. *El sistema permite seleccionar la fecha de vencimiento de la tarea.*
7. *El sistema muestra un formulario completo para la creación de una nueva tarea, este solicita más información sobre el evento que el formulario de Creación Rápida de Tarea.*
8. *El sistema permite guardar el registro de la nueva tarea.*
9. *El sistema permite cancelar la creación de la nueva tarea.*
10. *Todos los campos solicitados en este formulario son obligatorios ya que son los datos mínimos requeridos por el sistema para generar un nuevo evento, por esto son incluidos en la vista Creación Rápida de Tarea.*

Matriz de Componentes 0108
Nombre del Módulo o Vista: *Vista de Lista*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *El sistema presenta una lista de las actividades registradas en el calendario, las cuales pueden seleccionarse. La lista cuenta con diferentes herramientas y criterios de búsqueda.*
2. *El sistema permite seleccionar las acciones a realizar sobre las actividades mostradas en la lista y que hayan sido seleccionadas, estas acciones son: Cambiar Propietario, Eliminar, Importar, Exportar.*
3. *El sistema muestra el formulario extenso para ingresar una nueva tarea.*
4. *El sistema muestra el formulario extenso para ingresar un nuevo evento.*
5. *El sistema permite filtrar actividades por orden alfabético, según el asunto que se ha asignado a la actividad.*
6. *El sistema permite desplazarse por páginas a través del listado.*
7. *El sistema permite filtrar actividades por el estado que tengan (No iniciada, En Progreso, Completando, Esperando, Retrasado, Planeado, Realizada, Pendiente).*
8. *El sistema permite filtrar actividades por tipo (llamada, reunión, llamada móvil).*
9. *El sistema permite filtrar actividades por el asunto.*
10. *El sistema permite filtrar actividad por relación (En este momento el sistema no cuenta con información para mostrar opciones en este campo).*
11. *El sistema permite filtrar actividades por la fecha y hora de inicio.*
12. *El sistema permite filtrar actividades por la fecha de finalización.*

Para VTiger, inicialmente se habían considerado once vistas, sin embargo debido a la similitud entre algunas de ellas el trabajo de construcción de las matrices de componentes pudo ser sintetizado en seis matrices de componentes.

El análisis de este único pequeño segmento de la aplicación permite identificar un total de cincuenta y dos características relacionadas con el funcionamiento de la aplicación. Lo anterior da una idea del alto número de características con las que

cuenta un sistema funcional terminado. En la mayoría de los casos estas características no son tenidas en cuenta a la hora de realizar el diseño de una aplicación nueva y empiezan a ser identificadas durante la implementación generando cambios y demoras en las entregas.

6.2.2 Wordpad

Análisis realizado a la interfaz de inicio de Microsoft Wordpad

Primera Etapa: Registro de Parámetros Generales e Inventario de Interfaces

Tarea uno: Registro de Parámetros Iniciales

Figura 33. Formato parámetros iniciales Wordpad

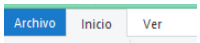
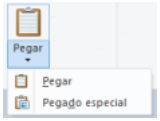
Formato Parámetros Iniciales	
Nombre de la aplicación: Wordpad	**Fecha:** 10/04/2015
Tipo de aplicación: Escritorio	
S.O sobre el cual se ejecuta: En este caso se está ejecutando sobre Windows 8. Pero se sabe que ha funcionado en muchas versiones anteriores de Windows.	
Propósito u objetivo de la aplicación: La aplicación permite realizar edición básica de texto.	
Quienes son los usuarios de la aplicación: Cualquier usuario del sistema operativo Windows.	
Forma en que se usa la aplicación: En este caso específico se está usando como una de las plataformas de prueba para la técnica ReFree.	
Dónde se usa la aplicación: Uso en un único computador personal.	
Número de Módulos para analizar: Uno (1) – Vista Principal	**Número de Formatos para analizar:** Uno (1)
Observaciones: Para efectos del desarrollo del ejemplo se tienen en cuenta solo algunas interfaces, estas están determinadas por el árbol de interfaces que se desarrolla más adelante.	

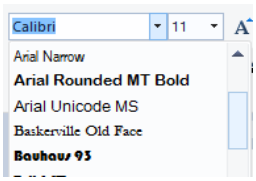
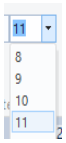
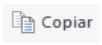
Tarea Dos: Inventario de Interfaces

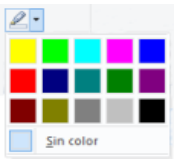
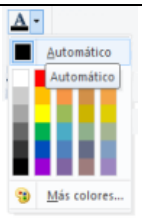
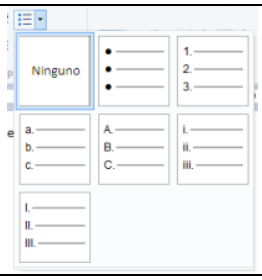
En este caso se selecciona una única vista a analizar, esta se denomina 00 Inicio y es vista principal de la aplicación.

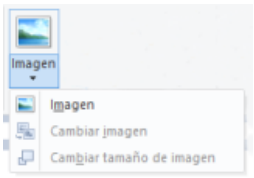
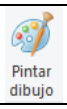
Segunda etapa: Construcción Matriz de Componentes

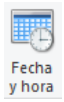
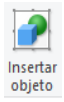
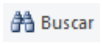
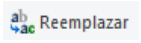
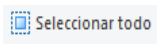
Matriz ID 00	**Nombre Modulo/Vista**	Vista Inicio Wordpad
Modelo de Interacción/ Patrón Lienzo y Paleta	**Descripción** Una paleta de íconos ubicada junto a un lienzo vacío, el usuario da clic sobre los botones de la paleta para crear objetos dentro del lienzo. La paleta por sí misma es una grilla de botones usualmente localizada en la parte superior izquierda del lienzo. Este patrón es usado en cualquier tipo de editor gráfico. En este caso es un editor de texto y las	

	herramientas están ubicadas en la parte superior dentro de un control tipo cinta.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Button		Guardar	Archivo	Guarda el documento de texto que está siendo creado o editado en el sistema.	"Guarda el documento activo"
Button		Deshacer	Acción	Elimina los efectos de la última acción realizada por el usuario en el sistema.	"Deshace la última acción"
Button		Rehacer	Acción	Revierte los efectos del componente anterior.	"Repite la última acción" Si el usuario utiliza la acción de deshacer una acción, este control permite rehacer inmediatamente la acción.
Ribbon Tabs		Seleccionar	Menú	Permite seleccionar tres conjuntos de controles diferentes, por defecto está seleccionada la pestaña de Inicio.	En este caso la interfaz de análisis es la de Inicio.
Button - Menu		Pegar	Objeto	Permite pegar/insertar elementos como texto, imágenes o tablas en el documento que está creando.	"Pega el contenido del portapapeles"
Button		Cortar	Objeto	Permite retirar elementos	"Corta la sección del lienzo y la

				como texto, imágenes o tablas en el documento que está creando.	coloca en el portapapeles”
Combo Box		Seleccionar	Fuente	Permite seleccionar el tipo de fuente que va a tener el texto del documento.	“Cambia la familia de fuentes”
Combo Box		Seleccionar	Tamaño Fuente	Permite seleccionar el tamaño de la fuente para el texto del documento.	
Button		Copiar	Objeto	Permite copiar elementos como texto, imágenes o tablas del documento que está creando.	“Copia la selección y pone en el portapapeles”
Button		Oscurecer	Fuente	Oscurece a negrilla la fuente del texto en el documento.	“Cambia a una fuente en negrilla”
Button		Inclinar	Fuente	Inclina la fuente del texto en el documento.	“Cambia a una fuente en cursiva”
Button		Subrayar	Fuente	Subraya la fuente del texto en el documento.	“Coloca una línea debajo del texto”
Button		Tachar	Fuente	Coloca una raya horizontal sobre la fuente del texto en el documento.	“Coloca una línea sobre el texto”
Button		Cambiar	Posición	Cambiar posición del texto para	“Dibuja el texto en menor tamaño y

				subíndice.	situado por debajo del texto normal”
Button		Cambiar	Posición	Cambiar posición del texto para superíndice.	“Dibuja el texto en menor tamaño y situado por encima del texto normal”
Button		Agrandar	Fuente	Permite hacer más grande la fuente del texto en el documento.	“Aumenta el tamaño de la fuente”
Button		Reducir	Fuente	Permite hacer más pequeña la fuente del texto en el documento.	“Reduce el tamaño de la fuente”
Combo Box		Seleccionar	Color	Permite seleccionar un color para resaltar partes del texto, simulando la acción de un marcador resaltador.	“Hace que el texto parezca resaltado con un marcador”
Combo Box		Seleccionar	Color	Permite seleccionar un color para la fuente del texto en el documento.	“Cambia el color del texto”
Button		Reducir	Sangría		“Reduce el nivel de sangría del párrafo”
Button		Aumentar	Sangría		“Aumenta el nivel de sangría del párrafo”
Combo Box		Seleccionar	Lista	Permite seleccionar un estilo para crear una lista en el documento.	“Hacer clic en la flecha para elegir estilos de lista diferentes”

Menu		Seleccionar	Interlineado	Permite seleccionar el espacio entre línea del texto. Y agrega o quita espacio entre los párrafos.	"Cambia el espaciado entre líneas de texto. Agrega o quita espacio después de los párrafos"
Button		Alinear Izquierda	Texto	Permite alinear el texto a la izquierda.	"Alinea el texto a la izquierda"
Button		Centrar	Texto	Permite centrar el texto.	"Centra el texto"
Button		A derecha	Texto	Permite alinear el texto a la derecha.	"Alinea el texto a la derecha"
Button		Justificar	Texto		"Alinea texto en los márgenes izquierdo y derecho agregando espacio adicional entre las palabras cuando sea necesario. De este modo se crea un aspecto ordenado a lo largo de los lados izquierdo y derecho de la página"
Button		Mostrar	Vista para configuración de párrafo.		"Muestra el cuadro de diálogo párrafo"
Button - Menu		Insertar	Imagen	Permite seleccionar una imagen desde los archivos para insertarla en el documento.	"Inserta una imagen de un archivo"
Button		Muestra	Vista del Paint	Inicializa el editor de imágenes	"Inserta un dibujo creado desde Microsoft

				Paint para construir la imagen que va a ser insertada en el documento.	Paint”
Button		Seleccionar	Formato de fecha y hora	Permite seleccionar el formato de fecha y hora que se quiere manejar en el documento.	“Hacer clic aquí para ver las opciones de formato de fecha y hora”
Button		Insertar	Objeto	Permite insertar objetos en el documento desde diferentes programas o archivos.	“Muestra el cuadro de diálogo insertar objeto”
Button		Buscar	Texto	Muestra un cuadro de diálogo, que permite ingresar el texto a buscar en el documento.	“Busca texto en el documento”
Button		Reemplazar	Texto	Muestra un cuadro de diálogo, que permite ingresar el texto a buscar y reemplazar en el documento.	“Reemplaza el texto en el documento.”
Button		Seleccionar	Texto, objetos.	Selecciona todo lo que se encuentre en el lienzo.	“Selecciona todo”
Nota:					

Tercera etapa: Traducción a Características

Matriz de Componentes 00
Nombre del Módulo o Vista: *Vista Inicio WordPad*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *El sistema guarda el documento que está siendo creado o editado.*
2. *El sistema elimina los efectos de la última acción realizada.*
3. *El sistema permite rehacer o repetir la última acción realizada.*
4. *El sistema permite seleccionar en un menú con varios conjuntos de herramientas, la vista principal y seleccionada por defecto está marcada con la palabra Inicio.*
5. *El sistema permite pegar objetos como texto imágenes o tablas en el documento, los cuales estén contenidos en el portapapeles.*
6. *El sistema permite cortar objetos como texto, imágenes o tablas y colocarlos en el portapapeles.*
7. *El sistema permite seleccionar la familia de fuente para el texto del documento.*
8. *El sistema permite seleccionar el tamaño de la fuente para el texto del documento.*
9. *El sistema permite copiar elementos como texto, imágenes o tablas del documento y colocarlos en el portapapeles.*
10. *El sistema oscurece la fuente del texto. Es decir cambiar la fuente a negrilla.*
11. *El sistema permite inclinar la fuente del texto. Esto es cambiar a una fuente cursiva.*
12. *El sistema permite subrayar la fuente del texto. Coloca una línea debajo del texto.*
13. *El sistema permite tachar la fuente del texto. Coloca una línea sobre el texto.*
14. *El sistema cambia de posición el texto. Dibuja el texto en menor tamaño y situado por debajo del texto normal para simular un subíndice.*
15. *El sistema cambia la posición del texto. Dibuja el texto en menor tamaño y situado por encima del texto normal para simular un superíndice.*
16. *El sistema permite agrandar el tamaño de la fuente del texto.*
17. *El sistema permite reducir el tamaño de la fuente del texto.*
18. *El sistema permite seleccionar un color para resaltar partes del texto, simulando la acción de un marcador resaltador.*
19. *El sistema permite seleccionar un color para la fuente del texto en el documento.*
20. *El sistema permite reducir la en nivel de sangría de los párrafos en el texto.*
21. *El sistema permite aumentar el nivel de sangría del párrafo.*
22. *El sistema permite seleccionar un estilo para crear una lista en el documento.*

23. *El sistema permite seleccionar el espacio entre línea del texto (interlineado). Además agrega y quita espacio entre los párrafos.*
24. *El sistema permite alinear el texto a la izquierda.*
25. *El sistema permite centrar el texto a la izquierda.6. El sistema permite alinear el texto a la derecha.*
26. *El sistema permite alinear el texto a la izquierda y a la derecha.*
27. *El sistema muestra el cuadro de diálogo de configuración de párrafo.*
28. *El sistema permite seleccionar una imagen desde los archivos para insertarla en el documento.*
29. *El sistema inicializa el editor de imágenes Paint de Microsoft para construir la imagen que va a ser insertada en el documento.*
30. *El sistema permite seleccionar el formato de fecha y hora que se quiere manejar en el documento.*
31. *El sistema muestra el cuadro de diálogo de insertar objeto, para insertar objetos en el documento desde otros programas o archivos.*
32. *El sistema muestra un cuadro de diálogo que permite ingresar el texto a buscar en el documento.*
33. *El sistema muestra un cuadro de diálogo, que permite ingresar el texto a buscar y reemplazar en el documento.*
34. *El sistema permite seleccionar todo lo que se encuentre en el lienzo.*

6.3 CASO SOFTWARE K2MEDICALWEB

En esta sección se presenta un caso de estudio mucho más amplio y perteneciente al entorno de software local, con las problemáticas reales latentes en los sistemas implementados y que encaja dentro de muchas de las consideraciones que se tuvieron en cuenta a la hora de plantear el presente trabajo. Cabe resaltar que fue una experiencia bastante gratificante poder encontrar todas estas características dentro de un sistema del mundo real, que está siendo utilizado y que requiere de una pronta intervención.

El sistema K2MedicalWEB, es un sistema de información para la gestión de servicios médicos actualmente utilizado en la Cruz Roja Colombiana Seccional Santander, gracias a la colaboración de la empresa MTI Manejo de Tecnologías de la Información S.A.S. quien para esta fecha se encarga de la gestión de sistemas en dicha institución fue posible llegar a conocer la situación de esta aplicación y las necesidades de la organización las cuales soportan en gran medida la necesidad de una herramienta como ReFree.

Las problemáticas que en este momento aquejan tanto a la aplicación K2MedicalWeb y a la organización en la cual esta implementada y en continuo uso son:

1. Ausencia de soporte por parte de los desarrolladores o distribuidores de la aplicación.
2. Algunos de los módulos de la aplicación están comenzando a presentar fallas.
3. Evolución en los procedimientos realizados por la organización lo que requiere la actualización del sistema.
4. Crecimiento de la organización (usuarios y clientes) requiere mejora en el manejo de los datos.
5. No es posible realizar copias de seguridad de los datos almacenados por el sistema de información.
6. La organización no cuenta con un acceso pleno a los datos almacenados, ni al diseño de la base de datos.
7. La organización no cuenta con fuentes de la aplicación.
8. El sistema parece estar siendo utilizado parcialmente debido a la imposibilidad de realizar ajustes y a la falta de conocimiento sobre sus características.
9. El sistema presenta comportamientos erráticos esporádicamente.
10. No existe ningún tipo documentación sobre descripción de la aplicación, configuración o diseño del sistema.
11. Usuarios no disponibles, no involucrados y en especial dispersos en distintas regiones del departamento.

A continuación se presentan las tres etapas desarrolladas en el análisis del K2MedicalWeb con ReFree, por efectos de espacio las matrices de componentes y la totalidad de los requerimientos recuperados se detallan en el anexo D del presente documento. Para la esta demostración se selecciona una sola matriz de componentes y se presentan los requerimientos hallados para esta matriz.

Primera Etapa: Registro de Parámetros Generales e Inventario de Interfaces

Tarea uno: Registro de Parámetros Iniciales

Figura 34. Formato parámetros iniciales K2MedicalWEB

Formato Parámetros Iniciales	
Nombre de la aplicación: K2MedicalWEB	**Fecha:** 16/09/2015
Tipo de aplicación: Web	
S.O sobre el cual se ejecuta: La aplicación está publicada en el servidor web IIS de un Windows Server 2008.	
Propósito u objetivo de la aplicación:* Administración de la información relacionada a los servicios médicos prestados por la Cruz Roja Colombiana en sus principales sedes en el departamento de Santander.	
Quienes son los usuarios de la aplicación:* Funcionarios del sector salud y administrativos vinculados con los procesos de la Cruz Roja Colombiana en la seccional Santander. Más específicamente médicos, enfermeras, contabilidad y administración.	
Forma en que se usa la aplicación:* Se usa desde consultorios y zonas de información. Los usuarios registran los datos de los pacientes. Los supervisores y coordinadores consolidan y analizan la información. Además se generan reportes a las entidades de vigilancia y control.	
Dónde se usa la aplicación:* La aplicación se unas en las seccionales de Bucaramanga y Barrancabermeja (Santander - Colombia).	
Número de Módulos para analizar:** 6 Módulos	**Número de Formatos para analizar:**** Cincuenta y dos (52)
Observaciones: El análisis de esta aplicación involucra los seis módulos principales presentados al inicio de la misma, para todos los módulos se llega a un primer nivel de profundidad y para el módulo de pacientes se analiza hasta el segundo nivel.	

Tarea dos: Inventario de Interfaces

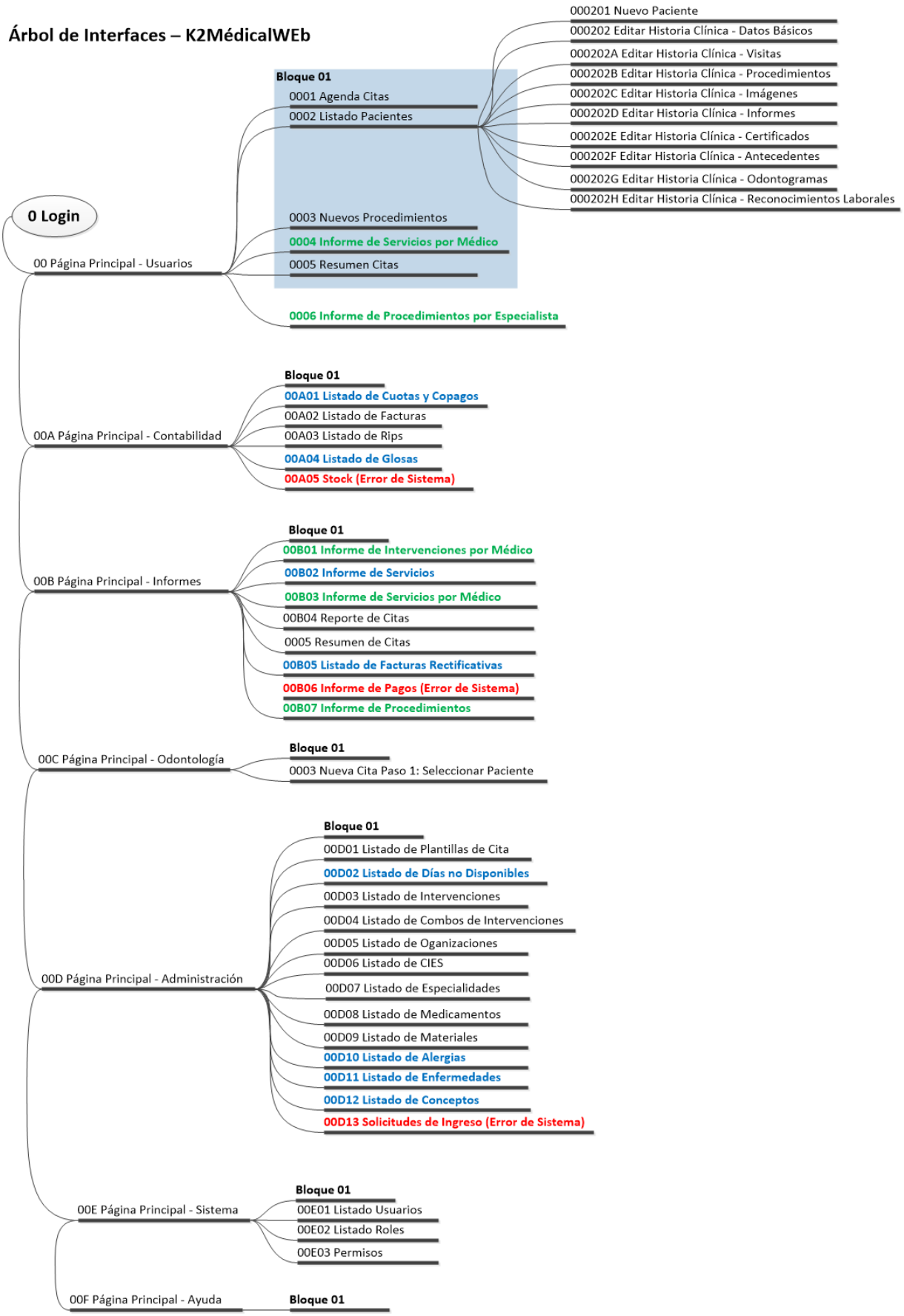
La tarea dos tiene como resultado el listado y el árbol de interfaces. La lista se presenta con una convención de colores simple que ayuda a identificar distintas situaciones que se presentaron durante la revisión de cada una de las interfaces (**Negro** Interfaces con funcionamiento normal. **Rojo** Interfaces con errores graves que no permiten la visualización. **Azul** Interfaces que no tenían información registrada asociada. **Verde** Interfaces que tienen acceso limitado por el tipo de usuario suministrado).

Tabla 17. Listado de interfaces K2MedicalWEB

0 Login	00 Página Principal - Usuarios	0001 Agenda Citas	
		0002 Listado Pacientes	Bloque 01
			000201 Nuevo Paciente
			000202 Editar Historia Clinica - Datos Básicos
			000202A Editar Historia Clinica - Visitas
			000202B Editar Historia Clínica - Procedimientos
			000202C Editar Historia Clínica - Imágenes
			000202D Editar Historia Clínica - Informes
			000202E Editar Historia Clínica - Certificados
			000202F Editar Historia Clínica - Antecedentes
			000202G Editar Historia Clínica - Odontogramas
			000202H Editar Historia Clínica - Reconocimientos Laborales
		0003 Nueva Cita Paso 1: Seleccionar Paciente	
		0004 Informe de Servicios por Médico	
		0005 Resumen Citas	
		0006 Informe Procedimientos por Especialista	
	00A Página Principal - Contabilidad	Bloque 01	
		00A01 Listado de Cuotas y Copagos	
		00A02 Listado de Facturas	
		00A03 Listado de Rips	
		00A04 Listado de Glosas	
		00A05 Stock (Error de Sistema)	
	00B Página Principal - Informes	Bloque 01	
		00B01 Informe de Intervenciones por Médico	
		00B02 Informe de Servicios	
		00B03 Informe de Servicios por Médico	
		00B04 Reporte de Citas	
		0005 Resumen de Citas	

		00B05 Listado de Facturas Rectificativas	
		00B06 Informe de Pagos (Error de Sistema)	
		00B07 Informe de Procedimientos	
	00C Página Principal - Odontología	Bloque 01	
		0003 Nueva Cita Paso 1: Seleccionar Paciente	
	00D Página Principal - Administración	Bloque 01	
		00D01 Listado de Plantillas de Cita	
		00D02 Listado de Días no Disponibles	
		00D03 Listado Intervenciones	
		00D04 Listado de Combos de Intervenciones	
		00D05 Listado de Organizaciones	
		00D06 Listado de CIES	
		00D07 Listado de Especialidades	
		00D08 Listado de Medicamentos	
		00D09 Listado de Materiales	
		00D10 Listado de Alergias	
		00D11 Listado de Enfermedades	
		00D12 Listado de Conceptos	
		00D13 Solicitudes de Ingreso (Error de Sistema)	
	00E Página Principal - Sistema	Bloque 01	
		00E01 Listado Usuarios	
		00E02 Listado Roles	
		00E03 Permisos	
	00F Página Principal - Ayuda	Bloque 01	

Figura 35. Árbol de interfaces K2medicalWEB



Segunda Etapa: Construcción Matriz de Componentes

Para esta etapa se presentan tres matrices, el total de cincuenta y dos matrices de componente analizadas para este proyecto se encuentran en el Anexo D del presente trabajo.

Tabla 18. Matriz de componentes 0 K2MedicalWEB

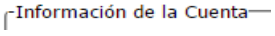
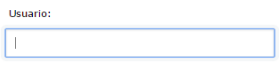
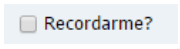
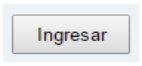
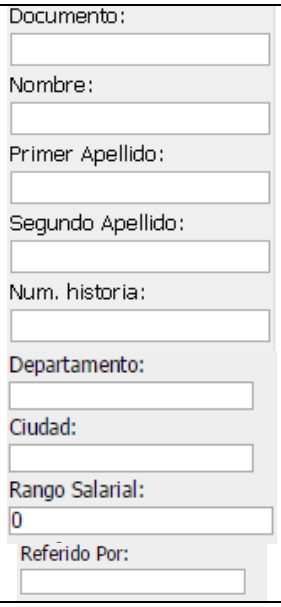
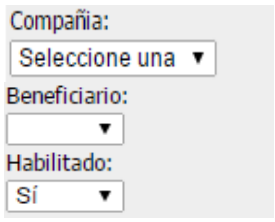
Matriz ID 0	**Nombre Modulo/Vista**	Login			
Modelo de Interacción/ Patrón	**Descripción** No se selecciona ningún patrón de interacción debido a la sencillez de la vista.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Image		Mostrar	Logos	Muestra el nombre de la aplicación y la institución en la que esté implementada.	Carácter informativo.
Label		Mostrar	Tipo de Información	La etiqueta indica el tipo de información que está siendo requerida por los campos que rodea.	Carácter informativo.
Text Box		Ingresar	Nombre	Espacio para ingresar el nombre de usuario.	
Text Box		Ingresar	Contraseña	Espacio para ingresar la contraseña	
Check Box		Seleccionar	Recordar Usuario	Permite seleccionar la opción de mantener la sesión iniciada.	
Button		Validar	Información	Realiza la validación de los datos ingresados en las cajas de texto por el usuario.	Si la información es válida el sistema muestra la vista 00 Página Principal
Nota: El sistema no cuenta con algún mecanismo de ayuda para recuperar la contraseña.					

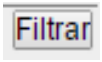
Tabla 19. Matriz de componentes 00E K2MedicalWEB

Matriz ID 00E	**Nombre Modulo/Vista**	Página Principal - Sistema			
Modelo de Interacción/ Patrón	**Descripción** No se selecciona ningún patrón de interacción debido a la sencillez de la vista. La vista cuenta con un conjunto de hipervínculos que permite navegar a otros lugares de la aplicación.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Link		Mostrar	Vista	El sistema cambia a la vista de listado de usuarios.	Vista desplegada: 00E01 Listado Usuarios.
Link		Mostrar	Vista	El sistema cambia a la vista de Listado de Roles.	Vista desplegada: 00E02 Listado Roles.
Link		Mostrar	Vista	El sistema cambia a la vista de Permisos.	Vista desplegada: 00E03 Permisos.
Nota: Los iconos mostrados a continuación, hacen parte de una barra de navegación global, la cual se encuentra presente en todas las vistas analizadas para este sistema por lo tanto no vuelven a repetirse en ninguna otra Matriz de Componentes.  Lo mismo para el ícono superior del sistema el cual regresa desde cualquier punto a la página de inicio y el link de usuario y cierre de sesión. 					

Tabla 20. Matriz de componentes 0002 K2MedicalWEB

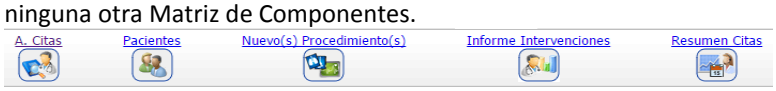
Matriz ID 0002	**Nombre Modulo/Vista**	Listado de Pacientes			
Modelo de Interacción/ Patrón Destacado, Búsqueda y Navegación	**Descripción** El sitio o aplicación contiene tres elementos obligatoriamente, un elemento destacado, una caja de búsqueda y una lista de elementos o categorías navegables. La aplicación posee una lista extensa de elementos que interesan al usuario, debe facilitar su búsqueda, a su vez alguno de estos elementos debe ser destacado ya que puede ser de interés para el usuario.				
Componente	**Imagen Componente**	**Verbo**	**Nombre**	**Función**	**Observaciones**
Button		Crear	Paciente	Al dar clic sobre este botón se despliega inmediatamente una nueva vista, la cual contiene un formulario con	Vista desplegada: 000201 Nuevo Paciente

				todos los datos requeridos para crear un nuevo paciente.	
Button		Deshabilitar	Paciente	Este control permite deshabilitar un paciente seleccionado de la lista de pacientes que se muestra en esta vista.	
Button		Habilitar	Paciente	Este control permite habilitar un paciente seleccionado de la lista de pacientes que se muestra en esta vista.	
Button		--	--	--	No realiza ninguna acción.
Text Boxes		Ingresar	Datos Filtro	El grupo de cajas de texto permite ingresar información necesaria para filtrar la búsqueda de un paciente en la lista mostrada.	No es obligatorio ingresar información en todas las cajas de texto.
Combos		Seleccionar	Datos Filtro	El conjunto de combos permite seleccionar información necesaria para filtrar la búsqueda de un paciente en la lista mostrada.	Compañía: No aparece ninguna. Beneficiario: Sí, No, todos. Habilitado: Sí, No, Todos.

Hipervínculo		Mostrar	Datos Filtro	La vista inicial muestra solo cinco datos de filtro, al dar clic en este hipervínculo se despliegan los otros datos para filtro que pueden ser utilizados.	Los demás controles para ingresar datos están simplemente ocultos dentro de la misma vista. El hipervínculo cambia a "MENOS FILTROS".
Button		Filtrar	Pacientes	Muestra en la lista de pacientes, aquellos que cumplen con los parámetros dados por el usuario en los Datos de Filtro.	Muestra dentro de la misma vista una lista filtrada.
Button		Quitar	Filtro	Retira todos los filtros colocados y muestra la lista de pacientes por defecto.	La lista por defecto muestra solo pacientes habilitados.
Check Box		Seleccionar	Paciente	Ubicada al lado izquierdo del nombre de cada uno de los pacientes de la lista, permite seleccionar al paciente para Deshabilitar o Habilitar,	
Link		Mostrar	Historia Clínica	El hipervínculo lleva a una nueva vista que muestra toda la información relacionada con el paciente para ser editada.	Vista Desplegada: 000202 Editar Historia Clínica - Datos Básicos Este hipervínculo está contenido en la tabla bajo el encabezado de Nombre.
Label Link	(*)	Desplazar	Lista	Los dos hipervínculos y la etiqueta configuran el	

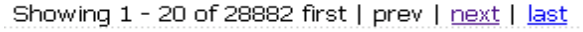
				desplazamiento por páginas de la lista de pacientes.	

Nota: Los iconos mostrados a continuación, hacen parte de una barra de navegación global, la cual se encuentra presente en todas las vistas analizadas para este sistema por lo tanto no vuelven a repetirse en ninguna otra Matriz de Componentes.



Lo mismo para el ícono superior del sistema el cual regresa desde cualquier punto a la página de inicio y el link de usuario y cierre de sesión.

(*) Imagen colocada aquí por el espacio.



Estructura de la lista de pacientes desplegada.

	Nombre	Num. Historia	Documento	Compañía	Teléfono	Beneficiario	Rango Salarial	Activo
☐	ACOSTA REY, MARIA DORA	2373	CC. 41603241	PARTICULARES	3118238811	No	0	Sí
☐	ALVARADO , LUIS EMILIO	941833	CC. 5727445	PARTICULARES	3164345037	No	0	Sí
☐	AMADOR VARGAS, OLIVARDO	942858	CC. 91243672	PARTICULARES	6576137	No	0	Sí

Tercera Etapa: Traducción a Características

Las características encontradas para las tres matrices anteriores se presentan a continuación, para las demás matrices de componentes las características encontradas se presentan en el Anexo E.

Matriz de Componentes 0
Nombre del Módulo o Vista: *Login*
Conjunto Principal de Características Candidato: *No definido*
Conjunto de Características Candidato: *No Definido*

1. *El sistema muestra los logos tanto del nombre de la aplicación como de la institución en la cual esta implementada. Esta información debe estar visible en todo momento durante el uso de la aplicación, en la parte superior de la interfaz.*
2. *El sistema muestra el tipo de información que requiere para realizar el ingreso a través de una etiqueta. En este caso marcada como "Información de la cuenta".*
3. *El usuario debe ingresar en el espacio dado el nombre de usuario.*
4. *El usuario debe ingresar en el espacio dado la contraseña de usuario.*
5. *El sistema permite al usuario seleccionar la opción de recordar la información de usuario suministrada.*
6. *El sistema evalúa los datos ingresados por el usuario para permitir o no el acceso del dicho usuario.*

Matriz de Componentes 00E
Nombre del Módulo o Vista: *Página Principal - Sistema*
Conjunto Principal de Características Candidato: *Administración Sistema*
Conjunto de Características Candidato: *Administrando Sistema*

1. *El módulo de administración del sistema muestra el listado de usuarios registrados en la aplicación.*
2. *El módulo de administración del sistema muestra el listado de roles que pueden tener los usuarios registrados en la aplicación.*
3. *El módulo de administración del sistema muestra el listado de permisos que pueden asignarse a los roles que tiene la aplicación.*

Matriz de Componentes 0002
Nombre del Módulo o Vista: *Listado Pacientes*
Conjunto Principal de Características Candidato: *Administración Pacientes*
Conjunto de Características Candidato: *Administrando Paciente*

1. *El sistema muestra una lista de todos los pacientes registrados en el sistema, esta lista cuenta con filtros de búsqueda que facilitan encontrar a determinado paciente dentro del sistema.*
2. *El sistema permite crear un paciente.*
3. *El sistema permite deshabilitar uno o varios pacientes seleccionados de la lista.*
4. *El sistema permite habilitar uno o varios pacientes seleccionados de la lista.*
5. *El sistema permite ingresar los siguientes datos para filtrar la búsqueda de pacientes: Documento, nombre, primer apellido, segundo apellido, número de historia, departamento, ciudad, rango salarial, referido por.*
6. *El sistema permite seleccionar otros datos de búsqueda, la compañía a la que está asociado el paciente, si este es beneficiario y si está habilitado o no.*
7. *El sistema filtra la búsqueda de pacientes de acuerdo a los datos suministrados.*
8. *El sistema quita los filtros y muestra por defecto la lista de todos los pacientes habilitados.*
9. *El nombre del paciente dentro de la lista se muestra como un link que lleva al usuario a la vista que permite editar la información del paciente. El formato del nombre es PRIMER APELLIDO SEGUNDO APELLIDO, NOMBRE.*
10. *La lista cuenta con una configuración de hipervínculos y etiquetas para poder desplazarse dentro de la lista cuando esta tiene una extensión considerable.*
11. *La lista se muestra ordenada alfabéticamente por apellidos.*

A través del análisis propuesto para el K2MedicalWEB no solamente se han identificado funciones que el sistema realiza, también se identifican errores y módulos que parecen no estar siendo utilizados dentro del sistema. Todos estos

elementos claves a la hora de realizar cualquier tipo de mantenimiento o rediseño de la aplicación. Y que en este caso son reportados para posibles decisiones organizacionales sobre el futuro de la aplicación.

Hay que resaltar que el interés del presente trabajo es la obtención de requerimientos en forma de una lista de características o descripciones en lenguaje natural del sistema analizado, esta lista tiene como objetivo primordial el poder conocer el dominio de la aplicación y la aplicación heredada de manera más autónoma y el poder realizar tareas de documentación, mantenimiento o rediseño.

La lista de características obtenida puede describirse como un registro en “bruto” de las funcionalidades del sistema, esto se hace evidente en la amplia cantidad de descripciones obtenidas tanto para las pruebas en el VTiger y el Wordpad como para el caso de estudio del K2MedicalWEB. En el caso de VTiger se analizan un total de once interfaces y se obtienen un total de cincuenta y dos características. El análisis del Wordpad se realiza sobre una sola interfaz y se obtienen treinta y cuatro características. Finalmente para el caso de estudio del sistema de información K2MedicalWEB se toman un total de cincuenta y dos interfaces, pertenecientes a los dos primeros niveles de profundidad y parcialmente el tercero, para este sistema se obtiene un total de doscientas cincuenta y cinco características. Esta amplia cantidad de información recuperada de manera organizada puede ser utilizada en próximas actividades del ciclo de vida del software.

No es el caso, para este trabajo el realizar la depuración y validación de estas descripciones. Esto depende más de las razones por las cuales un analista requiera la aplicación de ReFree y las herramientas con las que cuente para emprender esta tarea.

7. CONCLUSIONES Y TRABAJO FUTURO

Este proyecto logra proponer una técnica sencilla que permite extraer información de la interfaz gráfica de usuario de una aplicación heredada sin requerir acceso a su código fuente o bases de datos. Además la técnica ReFree presenta la información recuperada en un formato sencillo y en lenguaje natural que permite que esta sea entendida por los diferentes interesados que participan en el proyecto, especialmente cliente y usuarios.

Tanto en la investigación realizada previamente, como durante el desarrollo de la técnica se corrobora la necesidad de encontrar nuevas técnicas para la adquisición de requerimientos de software. Esto con el fin de mejorar y optimizar esta etapa del ciclo de vida del software.

A través de esta investigación se logra formalizar la observación y estudio de aplicaciones heredadas, las cuales en la práctica son hasta ahora actividades realizadas empíricamente por los desarrolladores. Estas actividades aún no se reconocen como una forma efectiva de aprovechar el software heredado como una fuente de información válida para el reconocimiento de un determinado dominio y entendimiento de las necesidades de los clientes y usuarios.

Teniendo en cuenta la definición de Ingeniería de Software del *Swebok - IEEE* como “(1) la aplicación de un enfoque sistemático, disciplinado y cuantificable del desarrollo, operación y mantenimiento de software, esto es la aplicación de ingeniería al software y (2) El estudio de los enfoques en (1).” La investigación realizada en el presente trabajo aporta elementos a la construcción de la ingeniería de software, ya que propone el desarrollo de un proceso sistemático, disciplinado y cuantificable para capturar requerimientos de software. Y realiza un estudio minucioso previo para proponer dicho enfoque.

El resultado que este trabajo entrega a la ingeniería de software, es un enfoque sistemático que encuentra una conexión entre controles y requerimientos, disciplinado porque consta de pasos plenamente definidos y repetibles y cuantificable ya que su resultado es un número determinado de requerimientos verificables de software.

A través de la hipótesis y el desarrollo de este proyecto se corrobora la existencia de prácticas dentro del desarrollo de software que nadie piensa que se puedan sistematizar. Pero que de manera metódica pueden convertirse en herramientas novedosas y útiles que aporten valor al ciclo de vida del software. Además de proporcionar una manera correcta y efectiva de hacer las cosas.

Como resultado extra en este proyecto de investigación se encuentra la efectividad del uso de una metodología aplicada a la construcción de ontologías para estructurar el desarrollo de un estado del arte. La adaptación de las tres

primeras etapas de Methontology permite delimitar el alcance del estado del arte y sintetizar la información recolectada a través de la búsqueda de documentación.

A través del estudio realizado en este trabajo, se encuentra la posibilidad de regresar al concepto de ingeniería inversa, el cual es una simple observación del artefacto analizado. Alejando su uso de la exclusiva aplicación de complicados algoritmos y técnicas orientadas más que a la observación a la ejecución automática de instrucciones, con resultados que en algunos casos se presentan en formatos de gran complejidad.

La información implícita dentro de las aplicaciones heredadas es una fuente de información importante a la hora de aprender sobre los dominios de dichas aplicaciones, es importante plantear nuevas alternativas que no impliquen complicados procesos a la hora de su implementación sino que al contrario encuentren en la simpleza de su desarrollo una manera de entender y percibir al software y a su construcción como una actividad que aunque ingenieril, va de la mano con las abstracciones, lenguaje y procesos cognitivos del ser humano.

Para darle continuidad a lo propuesto en este trabajo es posible pensar en el potencial de la presente técnica para presentar requerimientos en otro tipo de formato como Casos de Uso. E incluso contemplar la posibilidad de recuperar objetos y operaciones sobre dichos objetos. Los cuales son importantes elementos de diseño que pueden ser utilizados a la hora de realizar un mantenimiento o entender mejor el dominio de una aplicación.

En un futuro también se puede considerar el uso de técnicas como ReFree para reducir la necesidad de constante interacción con el usuario a la hora de iniciar un nuevo proyecto de software. Mejorando la calidad del tiempo que se cuenta para interactuar con el usuario ya que el diseñador conoce previa y ampliamente elementos del dominio a abordar.

Como proyectos a seguir a partir del presente trabajo, se puede considerar la necesidad de realizar un tipo de investigación cuantitativa, para medir la eficacia del uso de la técnica para disminuir el tiempo de interacción con el usuario. También es posible extender ReFree a otros tipos de aplicaciones como por ejemplo las desarrolladas actualmente para dispositivos móviles.

REFERENCIAS

[1] Chaos, “CHAOS MANIFESTO 2013: Think Big, Act Small,” *Standish Gr. Int.*, pp. 1–52, 2013.

[2] E. W. Dijkstra, “The humble programmer,” *Commun. ACM*, vol. 15, no. 10, pp. 859–866, 1972.

[3] ACSIS, “X Encuesta de Gerencia de Proyectos.” Asociación Colombiana de Ingenieros, Bogotá, 2012.

[4] I. Sommerville, *Ingeniería del Software*, 9th ed. México: Pearson Education, 2011.

[5] K. A. Briski, P. Chitale, V. Hamilton, A. Pratt, B. Starr, J. Veroulis, and B. Villard, “Minimizing code defects to improve software quality and lower development costs .,” *Development solutions*, no. October. IBM, p. 12, 2008.

[6] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 1st ed. England: Wiley, 2009.

[7] A. Finkelstein, “Bottom 10 Software Engineering Challenges,” 2013. [Online]. Available: http://blog.prof.so/2012/06/bottom-10-software-engineering.html. [Accessed: 20-Mar-2010].

[8] G. Pantaleo, *Calidad en el Desarrollo de Software*. Buenos Aires: Alfaomega, 2011.

[9] S. A. Fahmi and H.-J. Choi, “Software Reverse Engineering to Requirements,” *2007 Int. Conf. Converg. Inf. Technol. (ICCIT 2007)*, no. May 1998, pp. 2199–2204, Nov. 2007.

[10] K. Liu, A. Alderson, Z. Qureshi, and P. O. Box, “Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour Environment Observation of Human -Machine Interactions,” *Softw. Maintenance, 1999. (ICSM ’99) Proceedings. IEEE Int. Conf.*, pp. 3 – 12, 1999.

[11] M. Bano and D. Zowghi, “User involvement in software development and system success: a systematic literature review,” *… Eval. Assess. Softw. …*, pp. 125–130, 2013.

[12] A. Weitzenfeld, *Ingeniería de Software Orientada a Objetos con UML, Java e Internet*. México: Thompson, 2005.

[13] G. P. & D. W. Brett D. McLaughlin, *Head First Object-Oriented Analysis and*

Design, First. Sebastopol, CA: O'Reilly, 2006.

[14] P. P. Committee, *Guide to the Software Engineering Body of Knowledge 2004 Version SWEBOK ® A project of the IEEE Computer Society Professional Practices Committee*. 2004.

[15] J. Whitten L. and L. Bentley D., *Análisis de sistemas: diseño y métodos*, 3rd ed. México: McGraw - Hill, 2011.

[16] E. Hull, K. Jackson, and J. Dick, *Requirements Engineering*, 3rd ed. London: Springer, 2011.

[17] R. Lal, *Digital Design Essentials: 100 Ways to Design Better Desktop, Web and Mobile Interfaces*. Beverly, MA: Rockport Publishers, 2013.

[18] J. Tidwell, *Designing Interfaces*, 2nd ed. O'Reilly, 2011.

[19] R. S. Palmer and M. J. Felsing, *A Practical Guide to Feature-Driven Development*. Upper Saddle River: Prentice-Hall Inc., 2002.

[20] E. Elian, *Reversing: Secrets of Reverse Engineering*. Indianapolis, IN: Wiley, 2005.

[21] L. Fravre, *Model Driven Architecture for Reverse Engineering Technologies : Strategic Directions and System*. New York, New York, USA, 2010.

[22] O. Corcho, M. Fernández-lópez, and A. Gómez-pérez, "Building Legal Ontologies with METHONTOLOGY and WebODE," pp. 142–157, 2005.

[23] N. F. Noy and D. L. Mcguinness, "Desarrollo de Ontologías - 101 Guía Para Crear Tu Primera Ontología," pp. 1–29, 2005.

[24] E. Chikofsky and J. Cross, "Reverse Engineering and Design: A Taxonomy," *Software, IEEE (Volume7 , Issue 1)*, vol. 7, no. January, pp. 13–17, 1990.

[25] E. Merlo, J. F. Girard, I. Kontogiannisl, P. Panangaden, and R. De Mori, "Reverse engineering of user interfaces .*," in *Proceedings Working Conference on Reverse Engineering*, 1993, pp. 171 – 179.

[26] E. Merlo, "Reengineering User Interfaces," *Software, IEEE (Volume12 , Issue 1)*, no. January, pp. 64–73, 1995.

[27] W. Cohen, "Recovering software specifications with inductive logic programming," *AAAI*, 1994.

[28] J. Hainaut and V. Englebert, “Requirements for information system reverse engineering support,” *… Eng. 1995., …*, pp. 136–145, 1995.

[29] S. M. White, “Capturing requirements for legacy systems,” *Proc. 1995 Int. Symp. Work. Syst. Eng. Comput. Based Syst.*, pp. 251–256, 1995.

[30] M. M. Moore, “Rule - Based Detection for Reverse Engineering User Interfaces,” *Reverse Eng. 1996., Proc. Third Work. Conf.*, 1996.

[31] H. Dayani-Fard and I. Jurisica, “Reverse engineering by mining dynamic repositories,” *Proc. Fifth Work. Conf. Reverse Eng. (Cat. No.98TB100261)*, pp. 174–182, 1998.

[32] T. Richner and S. Ducasse, “Recovering high-level views of object-oriented applications from static and dynamic information,” *Softw. Maintenance, 1999.(ICSM’99) …*, 1999.

[33] E. Stroulia, L. Kong, P. Sorenson, A. B. Tg, and A. B. Te, “Reverse Engineering Legacy Interfaces : an Interaction-Driven Approach,” in *Reverse Engineering, 1999. Proceedings. Sixth Working Conference on*, 1999.

[34] M. El-Ramly, P. Iglinski, E. Stroulia, P. Sorenson, and B. Matichuk, “Modeling the system-user dialog using interaction traces,” *Proc. Eighth Work. Conf. Reverse Eng.*, pp. 208–217, 2001.

[35] E. Stroulia and T. Systä, “Dynamic analysis for reverse engineering and program understanding,” *ACM SIGAPP Appl. Comput. Rev.*, vol. 10, no. 1, pp. 8–17, Apr. 2002.

[36] M. El-ramly, E. Stroulia, P. Sorenson, A. Hall, and A. C. Tg, “Recovering Software Requirements from System-user Interaction Traces,” *SEKE ’02 Proc. 14th Int. Conf. Softw. Eng. Knowl. Eng.*, pp. 447–454, 2002.

[37] A. Memon, I. Banerjee, and A. Nagarajan, “GUI ripping: reverse engineering of graphical user interfaces for testing,” in *10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings.*, 2003, pp. 260–269.

[38] Y. Yu, Y. Wang, and J. Mylopoulos, “Reverse engineering goal models from legacy code,” in *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, 2005, pp. 363–372.

[39] T. Katsimpa, Y. Panagis, E. Sakkopoulos, G. Tzimas, and a. Tsakalidis, “Application modeling using reverse engineering techniques,” *Proc. 2006 ACM Symp. Appl. Comput. - SAC ’06*, p. 1250, 2006.

[40] S. Staiger, “Reverse Engineering of Graphical User Interfaces Using Static Analyses,” *14th Work. Conf. Reverse Eng. (WCRE 2007)*, pp. 189–198, Oct. 2007.

[41] D. Amalfitano, A. R. Fasolino, and P. Tramontana, “Reverse Engineering Finite State Machines from Rich Internet Applications,” *2008 15th Work. Conf. Reverse Eng.*, pp. 69–73, Oct. 2008.

[42] A. Jain and D. K. Tayal, “On reverse engineering an object-oriented code into UML class diagrams incorporating extensible mechanisms,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 33, no. 5, p. 1, Aug. 2008.

[43] M. Necasky, “Reverse Engineering of XML Schemas to Conceptual Diagrams`,” vol. 96, no. January, 2009.

[44] Ó. Sánchez Ramón, J. Sánchez Cuadrado, and J. García Molina, “Model-driven reverse engineering of legacy graphical user interfaces,” *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng. - ASE ’10*, p. 147, 2010.

[45] D. Amalfitano, A. R. Fasolino, and P. Tramontana, “An iterative approach for the reverse engineering of rich internet application user interfaces,” *5th Int. Conf. Internet Web Appl. Serv. ICIW 2010*, pp. 401–410, 2010.

[46] E. Shah and E. Tilevich, “Reverse-engineering user interfaces to facilitateporting to and across mobile devices and platforms,” *Proc. Compil. co-located …*, pp. 255–259, 2011.

[47] I. C. Morgado, A. C. R. Paiva, J. P. Faria, and R. Camacho, “GUI reverse engineering with machine learning,” *2012 1st Int. Work. Realiz. AI Synerg. Softw. Eng. RAISE 2012 - Proc.*, pp. 27–31, 2012.

[48] P. Aho, T. Räty, and N. Menz, “Dynamic reverse engineering of GUI models for testing,” *2013 Int. Conf. Control. Decis. Inf. Technol. CoDIT 2013*, pp. 441–447, 2013.

[49] R. J. Abbott, “Program design by informal English descriptions,” *Commun. ACM*, vol. 26, no. 11, pp. 882–894, 1983.

BIBLIOGRAFÍA

ABBOTT, Russell J. Program design by informal English descriptions. Communications of the ACM, 1983. pp. 882–894.

AHO, Pekka, RÄTY, Tomi y MENZ, Nadja. Dynamic reverse engineering of GUI models for testing. International Conference on Control, Decision and Information Technologies CoDIT, 2013. p. 441–447.

AMALFITANO, Domenico, FASOLINO, Anna Rita y TRAMONTANA, Porfirio. Reverse Engineering Finite State Machines from Rich Internet Applications. 15th Working Conference on Reverse Engineering, 2008. p. 69–73.

AMALFITANO, Domenico, FASOLINO, Anna Rita y TRAMONTANA, Porfirio. An iterative approach for the reverse engineering of rich internet application user interfaces. 5th International Conference on Internet and Web Applications and Services, 2010 p. 401–410.

ASOCIACIÓN COLOMBIANA DE INGENIEROS DE SISTEMAS ACIS. X Encuesta de Gerencia de Proyectos. Bogotá: ACIS, 2012.

BANO, Muneera, y ZOWGHI, Didar. User Involvement in Software development and System Succes: A Systematic Literature Review. IEEE Computer Society 2013.

BRISKI, Ann Kari et al. Minimizing code defects to improve software quality and lower development costs. United States of America: IBM Corporation, Development Solutions White Paper, 2008.

CHIKOFSKY, Elliot J. y CROSS II, James H. Reverse engineering and Design Recovery: A Taxonomy. IEEE, 1990.

COHEN, WW Recovering software specifications with inductive logic programming, AAAI, 1994.

CORCHO, Oscar, FERNÁNDEZ-LÓPEZ, Mariano y GÓMEZ-PÉREZ, Asunción. Building Legal Ontologies with METHONTOLOGY and WebODE, 2005. p. 142–157.

DAYANI-FARD, Homayoun, JURISICA, Igor. Reverse engineering by mining dynamic repositories, Proceedings Fifth Working Conference on Reverse Engineering (Cat. No.98TB100261), 1998. p. 174–182.

DIJKSTRA, Edsger. The Humble Programmer. En Classics in Software Engineering Editado por Edward Nash Yourdon. Nueva Jersey: Yourdon Press, 1979.

ELIAN, Eldan. Reversing: Secrets of Reverse Engineering. Indianapolis, IN: Wiley, 2005.

EL-RAMLY, Mohammad, et al. Modeling the system-user dialog using interaction traces. Proceedings Eighth Working Conference on Reverse Engineering, 2001.

EL-RAMLY, Mohammad, et al. Recovering Software Requirements from System-user Interaction Traces. Proceedings of the 14th international conference on Software engineering and knowledge engineering, 2002.

FAHMI, A. Syed y CHOI, Ho-Jin. Software Reverse Engineering to Requirements. IEEE Computer Society 2007.

FINKELSTEIN, Anthony. Bottom 10 Software Engineering Challenges. [Online]. Update June 23 de 2013. [Citado Ocutubre 2013] Disponible en: http://blog.prof.so/2012/06/bottom-10-software-engineering.html

FRAVRE, Liliana. Model Driven Architecture for Reverse Engineering Technologies : Strategic Directions and System. Nueva York, 2010.

HAINAUT, J-l, ENGLEBERT, V. Requirements for information system reverse engineering support, IEEE Engineering, 1995. p. 136–145.

HULL, Elizabeth, JACKSON, Ken y DICK, Jeremy, Requirements Engineering. 3 ed. Londres: Springer, 2011.

IEEE. Guide to the Software engineering body of knowledge, Swebook. Los Alamitos, California. IEEE Computer Society. 2004.

JAIN, Amita, TAYAL, Devendra K. On reverse engineering an object-oriented code into UML class diagrams incorporating extensible mechanisms. ACM SIGSOFT Software Engineering Notes, 2008.

KATSIMPA, T. et al. Application modeling using reverse engineering techniques. Proceedings of the 2006 ACM symposium on Applied computing - SAC ’06, 2006. p. 1250.

LAL, Rajesh. Digital Design Essentials: 100 Ways to Design Better Desktop, Web and Mobile Interfaces. Beverly, MA: Rockport Publishers, 2013.

LAMSWEERDE, A. van. Requirements Engineering: from system goals to UML models to software specifications. West Sussex: John Wiley and Sons, Ltd, 2009.

LIU, Kecheng, ANDERSON, Albert y QURESHI,Zubair. Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour. IEEE 1999.

MCLAUGHLIN, Brett D, POLLICE, Gary y WEST, David. Head First Object-Oriented Analysis and Design, First. Sebastopol, CA: O'Reilly, 2006.
MEMON, Atif, BANERJEE, Ishan y NAGARAJAN, Adithya. GUI ripping: reverse engineering of graphical user interfaces for testing. Proceedings 10th Working Conference on Reverse Engineering. 2003. p. 260–269.

MERLO, Ettore et al. Reverse engineering of user interfaces in Proceedings Working Conference on Reverse Engineering, 1993, p. 171 – 179.

MERLO, Ettore. Reengineering User Interfaces, Software, IEEE (Volume 12 , Issue 1), no. January, pp. 64–73, 1995.

MOORE, Melody M Rule - Based Detection for Reverse Engineering User Interfaces. Proceedings of the Third Working Conference on Reverse Engineering, 1996.

MORGADO, Inês Coimbra et al, GUI reverse engineering with machine learning. Proceedings 1st International Workshop on Realizing AI Synergies in Software Engineering, 2012. p. 27–31.

NECASKY, Martin. Reverse Engineering of XML Schemas to Conceptual Diagrams. Proceedings 6th Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), 2009.

NOY, Natalya F, MCGUINNESS, Deborah L. Desarrollo de Ontologías - 101 Guía Para Crear Tu Primera Ontología, 2005. p. 1–29.

PALMER, R. Stephen, FELSING, M. John. A Practical Guide to Feature-Driven Development. Upper Saddle River: Prentice-Hall Inc., 2002.

PANTALEO, Guillermo. Calidad en el desarrollo de software. 1a ed. Buenos Aires: Alfaomega, 2011.

RICHNER, Tamar DUCASSE, Stéphane. Recovering high-level views of object-oriented applications from static and dynamic information. Software Maintenance (ICSM'99), 1999.

SÁNCHEZ RAMÓN, Óscar, SÁNCHEZ CUADRADO, Jesús y GARCÍA MOLINA, Jesús. Model-driven reverse engineering of legacy graphical user interfaces. Proceedings of the IEEE/ACM international conference on automated software engineering - ASE '10, 2010. p. 147.

SHAH, Eeshan, TILEVICH, Eli. Reverse-engineering user interfaces to facilitate porting to and across mobile devices and platforms. Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11 - SPLASH '11 Workshops, 2011. p. 255–259.
SOMERVILLE, Ian. Ingeniería del Software. 9 ed. México: Pearson Educación, 2011. p. 773.

STAIGER, Stefan. Reverse Engineering of Graphical User Interfaces Using Static Analyses. 14th Working Conference on Reverse Engineering (WCRE 2007), 2007. p. 189–198.

STROULIA, Eleni, KONG, Lanyan, SORENSON, Paul. Reverse Engineering Legacy Interfaces : an Interaction-Driven Approach. Proceedings Sixth Working Conference on Reverse Engineering, 1999.

STROULIA, Eleni, SYSTÄ, Tarja. Dynamic analysis for reverse engineering and program understanding ACM SIGAPP Applied Computing Review, 2002.
THE STANDISH GROUP. The Chaos Manifesto. Boston: The Standish Group, 2013.

TIDWELL, Jenifer. Designing Interfaces, 2nd ed. O'Reilly, 2011.

WEITZENFELD, Alfredo. Ingeniería de Software Orientado a Objetos con UML, Java e Internet. México D.F. Thompson Learning 2005.

WHITE, S.M. Capturing requirements for legacy systems. Proceedings of the 1995 International Symposium and Workshop on Systems Engineering of Computer Based Systems, 1995. p. 251–256.

WHITTEN L. Jeffrey. BENTLEY D. Lonnie. Análisis de sistemas: diseño y métodos. 7 ed. Bogotá: McGraw-Hill, 2011.

YU, Yijun, WANG, Yiqiao y MYLOPOULOS, John. Reverse engineering goal models from legacy code. Proceedings. 13th IEEE International Conference on Requirements Engineering, 2005. p. 363–372.