MÉTODO DE VALIDACIÓN DE REQUISITOS FUNCIONALES DE SOFTWARE A PARTIR DE PROTOTIPOS DE INTERFACES DE USUARIO BASADOS EN PATRONES RIA

GERANA ESPINOZA CONTRERAS

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"

MÉTODO DE VALIDACIÓN DE REQUISITOS FUNCIONALES DE SOFTWARE A PARTIR DE PROTOTIPOS DE INTERFACES DE USUARIO BASADOS EN PATRONES RIA

Por:

GERANA ESPINOZA CONTRERAS

Trabajo presentado para optar
a la Categoría de Asistente en el escalafón
del Personal Docente y de Investigación

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA

Barquisimeto, Enero de 2013

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
COORDINACION DE POSTGRADO

ACTA VEREDICTO TRABAJO DE GRADO

Nosotros, Miembros del Jurado Examinador del Trabajo de Grado titulado: **"MÉTODO DE VALIDACIÓN DE REQUISITOS FUNCIONALES DE SOFTWARE A PARTIR DE PROTOTIPOS DE INTERFACES DE USUARIO BASADOS EN PATRONES RIA"**, presentado por **GERANA AUXILIADORA ESPINOZA CONTRERAS,** titular de la Cédula de Identidad **N° 12.724.615**, como requisito para optar al grado académico de **MAGÍSTER SCIENTIARUM EN CIENCIAS DE LA COMPUTACIÓN**, ofrecido por el programa de Maestría en Ciencias de la Computación del Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", hacemos constar que hoy catorce de noviembre del año dos mil doce (14/11/2.012) a las cuatro de la tarde (04:00 p.m.), se realizó el examen Público de Defensa de Trabajo de Grado, de acuerdo a lo establecido en la Normativa sobre Trabajos de Grado de la UCLA. Una vez rendido el examen, este Jurado emite el siguiente veredicto: El Trabajo de Grado fue:

**************** **APROBADO**
CON MENCIÓN HONORÍFICA ....

Dando fe de ello, levantamos la presente acta en la ciudad de Barquisimeto a los catorce días del mes de noviembre del año dos mil doce.

Prof. Edgar González
Presidente
C.I. Nº 7.311.123

Prof. Edison Sira
Tutor – Jurado Principal
C.I. Nº 7.367.942

Prof. Maritza Torres
Jurado Principal
C.I. Nº 7.372.706

DEDICATORIA

A Dios, por permitirme llegar a este momento tan importante, por los triunfos y los momentos difíciles que me han enseñado a valorarlo y amarlo cada día más.

A las personas más especiales y amadas de mi vida, mis padres, mi esposo y mis hijos.

AGRADECIMIENTO

A Dios, por darme salud, sabiduría, fortaleza y energía para culminar esta meta con gran éxito.

A mis queridos padres, Germán y Ana Josefina, que con su demostración de padres ejemplares me han enseñado a luchar hasta alcanzar mis metas. Gracias por sus sabios consejos, apoyo, comprensión, su inmenso amor y por todo lo que me han inculcado para llegar a ser la persona que soy.

A mi amado esposo Gianpiero, por su gran paciencia, apoyo, cariño, comprensión y amor incondicional, por siempre acompañarme durante todo este arduo camino. Te amo con todo mi corazón.

A mis hijos, Pedro y Germán, son el regalo más bello que me ha dado la vida, son la razón de mí existir, sin ellos la fuerza de levantarme cada día para ser mejor persona no sería una realidad. Espero que siempre se sientan orgullosos de mí.

A mis hermanos, Ana, Germán y Geranie, por estar siempre allí apoyándome, dándome fuerzas en todo momento, ustedes son parte fundamental de mi vida, los quiero muchísimo.

A mi suegra, Rosetta, por incentivarme siempre a seguir adelante, por su comprensión y paciencia, por estar siempre pendiente de mi y de mis hijos, por ser una segunda madre para mi, gracias!

A mi tutor el Prof. Edison Sira, por su apoyo incondicional en todo momento, por su atención y dedicación a mi trabajo, por ser un excelente tutor, muchísimas gracias.

A todos mis compañeros de maestría, gracias al equipo que formamos, ayudándonos y apoyándonos mutuamente, logramos llegar con éxito hasta el final del camino con mucha perseverancia y esfuerzo.

A mi amiga Desiree, por acompañarme, animarme y apoyarme en todo momento, especialmente en los más difíciles donde necesitaba una mano amiga. Gracias Desi.

A los profesores Edgar González, Jorge Pérez y Maritza Torres, gracias por su tiempo y apoyo, así como por la sabiduría que me transmitieron. Muchas gracias.

Gracias a todas las personas que de alguna u otra manera me ayudaron a lograr esta gran meta.

INDICE GENERAL

DEDICATORIA....iv
AGRADECIMIENTO.... v
INDICE DE ILUSTRACIONES....viii
INDICE DE TABLAS.... x
RESUMEN....xii
INTRODUCCIÓN.... 1
CAPÍTULO
I EL PROBLEMA.... 4
Planteamiento del Problema 4
Objetivos de la Investigación.... 17
Objetivo General: 17
Objetivos Específicos: 17
Justificación e Importancia 18
Alcance de la Investigación 21
II MARCO TEÓRICO 23
Antecedentes de la Investigación.... 23
Validación de Requisitos a través de Prototipos 24
Calidad del Software 34
Rich Internet Applications 38
Bases Teóricas 41
Ingeniería de Software.... 42
Ingeniería de Requisitos 43
Validación de Requisitos.... 49
Prototipos de Interfaz.... 50
Aplicaciones Web.... 51
RIA. *Rich Internet Application*.... 54
Patrones RIA 56
Modelo de Calidad de Software.... 63
Modelo McCall.... 64
Modelo ISO 9126 -1 66
Estándar ISO/IEC 25010 68
Fundamentos Metodológicos.... 71
Método de Investigación para la Ingeniería de Software.... 71
Ingeniería de Métodos 74
III MARCO METODOLÓGICO 80
Naturaleza de la Investigación.... 80
Diseño de la Investigación.... 81
Procedimiento de la Investigación 82
Documentación 82
Etapa 1. Determinación del Problema 83
Etapa 2. Creación de la Hipótesis 83

Etapa 3. Definición del Método de Trabajo 83
Etapa 4. Resolución, Validación y Verificación 84
IV PROPUESTA DEL ESTUDIO 87
Estructura de la Propuesta 88
Definición del flujo de trabajo del método VAREME 88
Definición de las métricas a utilizar en la evaluación de la calidad del prototipo basado en patrones RIA. 109
Método VAREME.....130
Validación del método VAREME ejemplificando su uso a partir de un caso de estudio. 137
V CONCLUSIONES Y RECOMENDACIONES 159
REFERENCIAS BIBLIOGRÁFICAS 163
ANEXO
A. Currículum Vitae del Autor 171

INDICE DE ILUSTRACIONES

Figura 1. Proceso de Ingeniería de Requisitos. 24
Figura 2. Flujo de trabajo del Subproceso de Validación de Requisitos. 25
Figura 3. Actividades que conforman el método propuesto por Ogata y Matsuura. 28
Figura 4. Cada usuario tiene su propio punto de vista. 30
Figura 5. Metodología de recopilación y validación de ideas y necesidades de los usuarios finales .. 32
Figura 6. Equivalencias McCall - ISO 9126 35
Figura 7. Mapa Conceptual Bases Teóricas 41
Figura 8. Actividades de Negociación de Requisitos y su Relación con la Clasificación. 47
Figura 9. Procesamiento de Datos en Aplicaciones Web Tradicionales 54
Figura 10. Procesamiento de Datos en Aplicaciones RIA. 56
Figura 11. Ejemplo Principio Make it Direct 57
Figura 12. Ejemplo Principio Keep It Lightweight 58
Figura 13. Ejemplo Principio Stay on the Page 59
Figura 14. Ejemplo Principio Provide an Invitation 60
Figura 15. Ejemplo Principio Use Transitions 61
Figura 16 .Ejemplo Principio React Immediately 62
Figura 17. Factores de Calidad de McCall 65
Figura 18. Modelo de calidad para calidad externa e interna del Estándar ISO/IEC 9126 68
Figura 19. Modelo de calidad para calidad externa e interna del Estándar ISO/IEC 25010 70
Figura 20. Método Genérico de Investigación propuesto por Marcos y Marcos. 73
Figura 21 Ejemplo de Process Deliverable Diagram (PDD) 78
Figura 22. Tipos de Actividades y Conceptos 79
Figura 23. Localización de la Investigación Científica y la Investigación Tecnológica en el continuo Investigar-Transformar. 81
Figura 24. Mapa Conceptual. Trabajos de Investigación tomados como base para la construcción del Método Propuesto. 88
Figura 25. Ejemplo de Diagrama de Procesos y Productos (PDD) 91
Figura 26. PDD del Subproceso de Validación de Requisitos. Método Gray Watch. 92
Figura 27. PDD del Método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web. 94
Figura 28. PDD del Proceso de Validación de Requisitos. Metodología Diseño del Pensamiento: escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios 97
Figura 29. Estructura Documento Requisitos de la Aplicación 103

Figura 30. PDD del Método VAREME 131
Figura 31. Actores del Sistema 139
Figura 32. Diagrama Casos de Uso Sistema Control de Ventas 139
Figura 33. Diagrama Casos de Uso Módulo Gestionar Artículos 140
Figura 34. Diagrama Casos de Uso Módulo Gestionar Clientes..... 140
Figura 35. Diagrama Casos de Uso Módulo Gestionar Ventas 141
Figura 36. Diagrama de Actividades del Escenario Registrar Ventas 143
Figura 37. Diagrama de Clases del Sistema Control de Ventas..... 144
Figura 38. Modelo de Navegación del Sistema Control de Ventas 145
Figura 39. Diagrama Entidad Relación del Sistema Control de Ventas..... 146
Figura 40. Base de datos relacional generada en MySQL del Sistema Control de Ventas 147
Figura 41. Pantalla de Autenticación de inicio al prototipo de la aplicación 148
Figura 42. Gestión de Clientes. Pantalla inicial (listado)..... 148
Figura 43. Pantalla de Edición de Clientes..... 149
Figura 44. Gestión de Artículos. Pantalla inicial (listado) 150
Figura 45. Gestión de Ventas. Pantalla inicial (listado) 150
Figura 46. Pantalla Reporte de Ventas Realizadas 151
Figura 47. Documento de Sesiones de Validación Establecidas 154
Figura 48. Gráfico escala de likert del cuestionario aplicado 157
Figura 49. Documento de Aceptación de Requisitos..... 158

INDICE DE TABLAS

Tabla 1. Cuadro de Resultados Obtenidos por los Estudios Realizados por Standish Group. 7
Tabla 2. Descripción de Flujo de Trabajo: Validación de Requisitos 25
Tabla 3. Resumen de Montilva y otros (2008) 26
Tabla 4. Resumen de Ogata y Matsuura (2010) 29
Tabla 5. Resumen de Gabrysiak y otros (2011) 33
Tabla 6. Factores para Evaluar la Calidad del Software por parte del Usuario 36
Tabla 7. Resumen de Macías y Gómez (2010) 37
Tabla 8. Resumen de Scott y Neil (2009) 39
Tabla 9. Enfoques para la Creación de Métodos en la Ingeniería de Métodos Situacional. 76
Tabla 10. Actividades del Subproceso de Validación de Requisitos. Método Gray Watch 93
Tabla 11. Actividades del Método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web. 95
Tabla 12. Actividades de la Metodología Diseño del Pensamiento: Escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios. 98
Tabla 13. Comparación Fases de los Métodos pertenecientes al Método Base 100
Tabla 14. Matriz Comparación Métodos 101
Tabla 15. Características de Calidad del Modelo ISO/IEC 25010 usados para Evaluar la Calidad por parte del Usuario Final, del Prototipo de Interfaces de Usuario basado en Patrones RIA del Método VAREME 111
Tabla 16. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica “Facilidad de Entendimiento”. 116
Tabla 17. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica “Facilidad de Aprendizaje”. 117
Tabla 18. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica “Operabilidad”. 118
Tabla 19. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica “Protección contra errores del usuario”. 119
Tabla 20. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica “Estética de la Interfaz de Usuario”. 119
Tabla 21. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a las característica Funcionalidad, Confiabilidad y Eficiencia 120
Tabla 22. Cuestionario de Evaluación de Calidad del Prototipo de Interfaces de Usuario basado en Patrones RIA del Método VAREME 121

Tabla 23. Patrones RIA recomendados usar, asociados a los factores de calidad identificados para medir la característica Usabilidad del modelo ISO/IEC 25010 126
Tabla 24. Cuestionario de Evaluación del Uso de Patrones RIA recomendados en el Prototipo de Interfaces de Usuario a Utilizar en el Método VAREME 129
Tabla 25. Actividades y Sub-Actividades del Método VAREME 132
Tabla 26. Productos del Método VAREME 135
Tabla 27. Descripción Caso de Uso Registrar Venta 142
Tabla 28. Cuestionario de Evaluación del Uso de Patrones RIA aplicado al Prototipo del Caso de Estudio presentado 152

UNIVERSIDAD CENTROCCIDENTAL LISANDRO ALVARADO
DECANATO DE CIENCIAS Y TECNOLOGÍA
COORDINACIÓN DE POST-GRADO

MÉTODO DE VALIDACIÓN DE REQUISITOS FUNCIONALES DE SOFTWARE A PARTIR DE PROTOTIPOS DE INTERFACES DE USUARIO BASADOS EN PATRONES RIA

Autor: Gerana Espinoza Contreras
Tutor: Edison Sira Carreño

RESUMEN

La aplicación de la Ingeniería de Requisitos en forma apropiada aumenta las posibilidades de producir software que satisfaga las necesidades de los usuarios, muchos errores de la etapa de elicitación de requisitos tienen su origen en la ambigüedad presentada entre usuarios finales y desarrolladores. Es por ello que se recurre a la validación de los requisitos, y al uso de técnicas como los prototipos de interfaces de usuario, que en corto tiempo, proporcionen al usuario una visión preliminar del sistema futuro. Con la presente investigación se diseñó el método VAREME, cuyo propósito es permitir al ingeniero de software validar requisitos funcionales de software a partir de prototipos de interfaces de usuario, usando para ello un enfoque enmarcado en métricas de calidad e incorporando los patrones de interfaz humano-computador *Rich Internet Application*s (RIA). La investigación se desarrolló bajo la modalidad de proyecto especial con un apoyo en Ingeniería de Métodos Situacional, tomando un enfoque basado en ensamblaje que permite construir un nuevo método a partir de fragmentos de otros métodos existentes. En los resultados se destaca el uso del metamodelo PDD, una guía de patrones RIA, y la incorporación de métricas de calidad adaptadas al modelo ISO/IEC 25010 evaluadas con instrumentos asociados a los prototipos que resulten al aplicar el método propuesto.

Palabras Clave: Validación de Requisitos, Prototipos, Interfaces de Usuario, RIA, Calidad del Software.

INTRODUCCIÓN

Recientemente, la Ingeniería de Requisitos ha cobrado una importancia cada vez mayor en la academia y en la industria, debido a que se espera que los productos de software proporcionen funciones centradas en el usuario de mayor calidad y seguridad (Westfall, 2011). Buena parte de los problemas que surgen a lo largo del proceso de desarrollo de software se deben a la carencia de un proceso adecuado de definición y entendimiento de los requisitos y del problema a resolver, y a la interpretación poco clara de las necesidades del cliente. Es por ello que la gestión de requisitos en la Ingeniería de Software, es una de las principales estrategias para garantizar la calidad de las aplicaciones desde las primeras etapas del proceso de desarrollo de software.

Los errores en la etapa de elicitación de requisitos generalmente se deben a que el ingeniero de software no esta completamente seguro de haber entendido correctamente los requisitos obtenidos, o que algunos de ellos no estén demasiado claros, con lo cual es de vital importancia la validación de requisitos. Muchas veces los requisitos tienden a descubrirse a medida que el usuario se familiariza con las tecnologías de información y con el propio sistema, por ello es de gran utilidad el uso de técnicas como la elaboración de prototipos de interfaces de usuario que permitan aclarar y asistir el panorama a los usuarios en cuanto a la elicitación y validación de los requisitos funcionales del mismo, de esta forma el usuario puede comenzar a experimentar la interacción con un elemento dinámico que le ayude a ir refinando sus exigencias y expectativas para el sistema que está naciendo (Pressman, 2010).

En este sentido, según Meixner, Paternò y Vanderdonckt (2011)**,** es importante destacar el gran significado que tienen las interfaces de usuario en los sistemas informáticos. El nexo de unión de todo sistema con sus potenciales usuarios es la interfaz humano-computador que la aplicación les ofrece, por lo que hay que considerar que un sistema es una herramienta útil si el usuario final puede utilizarla con facilidad. El proceso de interacción humano-computador determina la valoración

que tiene el usuario de un sistema. De este manera, los usuarios asignan una buena calidad al software si hace lo que ellos desean, en una forma que sea fácil de aprender y fácil de utilizar, ya que ellos son quienes realmente saben cuáles son sus necesidades y los resultados que esperan obtener.

Shneiderman y Plaisant (2010), afirman que el éxito de un sistema de información, más que depender de un nivel de calidad evaluado internamente, se relaciona con el grado de aceptación y satisfacción del usuario que, no sólo se conforma con que el software cumpla con los requisitos funcionales, sino que le otorga gran importancia a la interfaz de usuario. Como consecuencia directa, las interfaces de usuario entregadas en productos finales están cada vez mejor trabajadas, presentadas y adecuadas, para ofrecer al usuario sensaciones de seguridad, fiabilidad, sencillez y facilidad de uso, permitiéndole manejar de forma natural e intuitiva la aplicación, logrando de esta manera su satisfacción.

Por otra parte, hoy en día la tendencia mayoritaria en el desarrollo del software es el empleo de tecnología Web, debido al crecimiento desenfrenado que está teniendo la la misma, se esta convirtiendo en parte integral de la vida diaria de cualquier individuo y en un elemento fundamental de la sociedad. Es por ello que las aplicaciones Web juegan un papel fundamental dentro de este contexto, siendo un desafío para los ingenieros de software garantizar su evolución continua, la inmediatez de su desarrollo, la seguridad en su operación y la alta demanda estética, así como la entrega de contenido funcional, que son factores que las caracterizan (Pressman, 2010).

Por lo tanto, el hecho de que la Web esté siendo cada vez más utilizada para nuevos propósitos y por todo tipo de usuarios, lleva a la necesidad de reconsiderar la forma en que las interfaces deben ser creadas, desarrollando aplicaciones Web orientadas al usuario final, el cual es el objetivo de la Web 2.0, que es la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones web enfocadas al usuario final que exigen una alto grado de facilidad de uso y gran alcance entre acciones (O'Reilly y Battelle, 2009).

De este modo todas las características tecnológicas de la Web 2.0 se han concentrado en lo que se conoce como Rich Internet Applications (RIA) las cuales están orientadas a satisfacer las expectativas del usuario en términos de usabilidad, funcionalidad, fiabilidad, facilidad de mantenimiento y rendimiento. Las RIA combinan la interactividad y las funcionalidades de las interfaces de una aplicación de escritorio común con la arquitectura de distribución ligera de la Web obteniendo como resultado una mejora en todos los elementos de una aplicación Web, incrementando la satisfación del usuario (Rossi, Fraternali y Sánchez, 2010).

De lo anteriormente descrito, surge la presente investigación, que tiene como objetivo principal diseñar un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones Rich Internet Applications (RIA), bajo un enfoque de calidad. El estudio se encuentra dividido en tres (3) capítulos los cuales se detallan a continuación:

En el Capítulo I, se plantea el problema de la investigación, se define el objetivo general y los específicos, justificación e importancia y los alcances de la investigación.

El Capítulo II, reseña las investigaciones anteriores que sirven de antecedentes a la investigación, comentando su contenido y aporte realizado. También se desarrolla el fundamento teórico que respalda este estudio.

Adicionalmente en el Capítulo III, se realiza una ubicación del tipo y la modalidad de la investigación a partir de criterios aceptados, seguidos por las fases que serán desarrolladas durante el proceso de investigación para dar respuesta a los objetivos planteados.

El Capítulo IV, contiene la propuesta del estudio, se presentan los pasos a seguir para construir el método de validación de requisitos funcionales de software a partir de prototipo de interface de usuario basado en patrones RIA, bajo un enfoque de calidad. También se presenta un caso de estudio para validar el uso del método propuesto.

El Capitulo V, presenta un conjunto de conclusiones obtenidas del estudio realizado y algunas recomendaciones para futuras investigaciones. Finalmente se presentan las Referencias Bibliográficas.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

En la actualidad, los sistemas de información se caracterizan por su complejidad y dinamismo, jugando un papel fundamental para el éxito de las empresas, de allí la importancia de diseñar y desarrollar sistemas, software, de calidad que satisfagan las necesidades del cliente final. (Kalareh, 2012). Por lo que, dentro del proceso de desarrollo de software se identifican diversos aspectos muy relevantes y necesarios para el desenvolvimiento formal de un proyecto que lo dirija hacia el éxito en esta disciplina. En tal sentido, se deben atender en profundidad elementos como la definición de actividades a realizar, utilización de metodologías para apoyar directamente los procesos de planificación y desarrollo de software, gestión de requisitos, pruebas de software, entre otros. Es por ello, que Sommerville (2005) define la Ingeniería de Software como una disciplina de la ingeniería que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación de requisitos del sistema, hasta el mantenimiento del mismo después de la puesta en marcha y la utilización.

Es importante resaltar que la noción de Ingeniería de Software fue propuesta inicialmente en 1968, en una conferencia sobre desarrollo de software para plantear la problemática que ofrecía el software en ese momento, a lo que se le llamo “crisis mundial del software”. Los grandes proyectos generalmente tenían años de retrasos, costaban más de lo presupuestado, eran irrealizables, con un desempeño deficiente,

poco fiables y difíciles de mantener, producto del crecimiento espectacular de la demanda de sistemas de computación cada vez más y más complejos, asociado a la inmadurez del sector informático, a la falta de recursos, a la escasez de métodos, normativas y estándares, que pudieran aclarar el panorama en cuanto a la visión de un proyecto enmarcado en la disciplina de ingeniería de software, la cual estaba en sus inicios. Por lo tanto se necesitaban nuevas técnicas y métodos para controlar la complejidad asociada a los grandes sistemas (Joyanes, 1996).

Desde entonces hasta el presente, se han realizado enormes progresos, apareciendo herramientas, métodos y técnicas, que forman parte de la ingeniería de software, presentándose así, como la solución definitiva al problema de la planificación, previsión de costes, reducción de esfuerzo y aseguramiento de la calidad en el desarrollo de software. Por otra parte, Pressman (2010) sostiene que el objetivo primordial de la ingeniería del software es producir un sistema, aplicación o producto de alta calidad, a través de métodos efectivos junto con herramientas modernas dentro de un proceso maduro de desarrollo de software. De allí la necesidad de usar y crear métodos cuyo propósito sean facilitar la producción de software de alta calidad.

De acuerdo a Arias (2005), uno de los principales problemas en el desarrollo de sistemas grandes y complejos, es el de la elicitación o descubrimiento de requisitos funcionales y no funcionales que de alguna manera van a incidir en la construcción del software, afirmando que este proceso es una de las piezas fundamentales en un proyecto de desarrollo de software, ya que marca el punto de partida para actividades como la planificación, en lo que se refiere a las estimaciones de tiempos y costos y la elaboración de cronogramas durante la etapa de desarrollo.

Por otra parte, una investigación realizada por Rodrigues (2001), encontró que los proyectos de desarrollo de software son de tiempo comprimido, intensivo, y de misión crítica, donde los requisitos son pobremente definidos. Los encuestados mencionaron varios problemas que impiden la entrega oportuna de los proyectos de software, los

cuales se mencionan continuación[1]:

- Requisitos inestables, en constante cambio (66%).
- Especificación de los requisitos pobre y deficiente (55%).
- El comportamiento del cliente, tales como los retrasos de aprobación, cambios en los requisitos, y la falta de comunicación (42%).

Como se puede apreciar, las etapas de la elicitación, especificación y la gestión del proceso de requisitos, participaron en casi todas estas causas. La falta de cuidado en la comprensión, la documentación y la gestión de requisitos puede llevar a una gran cantidad de problemas: construir un sistema que resuelve el problema equivocado, que no funciona como se espera, o que presenta dificultades para que los usuarios puedan comprenderlo y utilizarlo.

En este orden de ideas, The Standish Group[2], realiza un estudio de proyectos de desarrollo de software aproximadamente cada dos (2) años, en diferentes organizaciones públicas y privadas en Estados Unidos y Europa, denominado Chaos Report, cuyo objetivo es documentar los fracasos en el desarrollo de iniciativas en tecnologías de información, y encontrar sus principales causas para tratar de evitarlas. En su reporte del año 2009, denominado "Chaos Report 2009" se estudiaron varios proyectos (no especifican cuantos), donde se obtuvieron los siguientes resultados (The Standish Group, 2009):

- Solo el 32% de los proyectos de software tuvieron éxito.
- El 44% de los proyectos de software fueron problemáticos, es decir, tuvieron retrasos, gastaron más dinero de lo presupuestado, o no todas las funcionalidades requeridas pudieron terminar operativas.
- El 24% de los proyectos de software terminaron en fracaso, ya que se cancelaron antes de su implantación o nunca fueron utilizados.

---

1 Los porcentajes resultantes de la investigación realizada se encuentran solapados.

2 The Standish Group es una empresa consultora en proyectos tecnológico que se encarga de recopilar información sobre los fracasos en la Industria de la Tecnología de Información, http://www.standishgroup.com

De este modo, según Domínguez (2009), el director de Informática de la empresa The Standish Group, aseguró que estos números representan el porcentaje más alto de fracaso en una década. Esta situación se puede notar en el siguiente cuadro comparativo donde se detallan los resultados obtenidos en los estudios realizados por la empresa en los últimos años:

Tabla 1. Cuadro Comparativo de Resultados Obtenidos por los Estudios Realizados por Standish Group.

Clasificación de Proyectos	1994	1996	1998	2000	2002	2004	2006	2009
Exitosos	16%	27%	26%	28%	34%	29%	35%	32%
Cuestionables	53%	33%	46%	49%	51%	53%	46%	44%
Fracasados	31%	40%	28%	23%	15%	18%	19%	24%

Fuente: Domínguez (2009)

Así, para comprender el por qué de estos resultados, The Standish Group pidió a los participantes en el estudio que explicaran las causas de los proyectos fallidos. Los principales factores fueron:

- 13% Requisitos Incompletos.
- 12% Falta de Compromiso del Usuario.
- 11% Recursos Inadecuados.
- 10% Expectativas no realistas.
- 9% Falta de Soporte Gerencial.
- 9% Requisitos Cambiantes.
- 8% Planeamiento Inadecuado.

Del mismo modo, Westfall (2011) llevo a cabo una investigación con el objetivo de conocer el estado de la práctica de la Ingeniería de Requisitos en diferentes organizaciones, tales como universidades, empresas privadas ubicadas en Colombia, y corporaciones multinacionales cuyas sedes se encuentran en los EE.UU., Japón,

Suecia, Alemania, Finlandia y la India. La investigación se realizó en el año 2010 y consistió en aplicar una encuesta en donde se les pedía a los participantes responder un cuestionario en línea a través de una página Web, sobre datos básicos de la organización a la pertenecen, su práctica regular asociada a la Ingeniería de Requisitos, en particular, acerca de los enfoques de elicitación y las técnicas de representación de requisitos.

La base de la muestra incluyó a 377 profesionales, procedentes de 237 compañías de software u organizaciones de investigación. Las áreas de negocio de las empresas involucradas cubren diversos segmentos de la industria, como la electrónica y la robótica, entre otros. Todos los participantes en la encuesta son profesionales de software cuyo trabajo diario está íntimamente relacionado con la Ingeniería de Requisitos. En particular, se buscó que los entrevistados narraran, mediante historias, lo qué han aprendido de sus éxitos y fracasos en la Ingeniería de Requisitos.

Dichos resultados mostraron que el estado de la práctica de la Ingeniería de Requisitos varía de una organización a otra, dependiendo del nivel de conocimientos técnicos y la experiencia en el campo, sin embargo se obtuvo como resultado un mayor porcentaje de experiencias de fracaso que de éxito, concluyendo que las razones más importantes por las cuales falla la práctica de la Ingeniería de Requisitos son las siguientes (Westfall, 2011):

- Los clientes no tienen una comprensión clara de los requisitos del sistema, incluyendo el alcance del mismo, las principales características funcionales y los atributos no funcionales.
- Las necesidades y la comprensión de los usuarios cambia constantemente.
- Los ingenieros de software no tienen suficiente acceso al conocimiento y a la experiencia del dominio.
- El cronograma del proyecto es demasiado estrecho como para permitir una adecuada interacción y un periodo de aprendizaje entre el cliente y el equipo de desarrollo.
- La reutilización de diseños existentes tiene un contexto y un entorno

equivocados.

- A los tomadores de decisiones en requisitos les falta conocimientos técnicos y del dominio.
- Faltan vínculos de comunicación entre clientes, analistas y desarrolladores.
- Falta un dominio estandarizado de definición de datos y de interfaz del sistema y del entorno.

De esta manera, estos estudios presentados ponen de manifiesto el hecho de que, a pesar de que las herramientas para construir software han evolucionado enormemente, se sigue produciendo software que no es satisfactorio para los clientes y usuarios, y los principales problemas residen en las primeras etapas del desarrollo, cuando hay que decidir las características del producto software a desarrollar, de allí la importancia de la Ingeniería de Requisitos.

Montilva, Barrios y Rivero (2008), afirman que la Ingeniería de Requisitos surge como una disciplina dentro de la Ingeniería de Software encargada del estudio de los requisitos en los sistemas automatizados, estableciendo principios, modelos, métodos, técnicas y herramientas automatizadas que contribuyen a mejorar la definición y especificación de requisitos, lo cual permite lograr un entendimiento común entre clientes, usuarios e ingenieros de software sobre los requisitos que debe satisfacer la aplicación de software. Es por ello que, el uso de métodos, herramientas y prácticas adecuadas en la gestión de requisitos de software, es una condición fundamental para lograr productos de calidad (Anaya, Londoño y Hurtado, 2006).

Se podría decir que el desarrollo de software es un proceso complejo y entregar un producto de calidad que cumpla con las expectativas del cliente puede ser un proceso complicado. Según Pressman (2010), afirma que por lo general los clientes no saben exactamente lo que desean y muchas veces los requisitos del software no reflejan las necesidades reales de los mismos y del ambiente organizacional; por lo cual resulta necesario validar los requisitos obtenidos con los usuarios una vez que haya terminado el proceso de elicitación, o a través de un proceso continuo incremental

alineado con las metodologías de ingeniería de software que se estén aplicando.

En este orden de ideas, el concepto de validación en la Ingeniería de Software, de acuerdo a Pressman (2010), se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente; por otro lado, Bohem (1981) lo define en base a la siguiente interrogante: ¿Estamos construyendo el producto correcto? Así, Pressman (2010) define validación de requisitos como la actividad de la Ingeniería de Requisitos en la que clientes y usuarios, junto con la ayuda de los ingenieros de requisitos, examinan las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto que se está desarrollando.

En este sentido, Sommerville (2005) indica que una técnica que ayuda enormemente a validar los requisitos obtenidos con los usuarios, es la elaboración de prototipos de interfaz, que ofrezcan a los usuarios una forma rápida de tener una visión preliminar del sistema final. Así, Arias (2005) define prototipos como simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requisitos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva.

De manera formal, la IEEE[3] *(Institute of Electrical and Electronics Engineers*), señala que la técnica de prototipos o prototipaje es usada frecuentemente durante las fases que involucran la elicitación, especificación y validación de los requisitos de un proyecto, ya que el cliente puede reaccionar mejor al ver un prototipo que al leer una especificación de requisitos logrando un feedback más rápido. El prototipo permite definir el alcance en el proceso de especificación de requisitos. Un documento de

[3] IEEE: Instituto de Ingenieros Eléctricos y Electrónicos, http://www.ieee.org/index.html

especificación de requisitos de software apoyado en un prototipo, tiende a sufrir menos cambios durante la fase de desarrollo acortando el tiempo empleado en esto y reduciendo los costos. Así el uso de prototipos facilita los proceso de descubrimiento y especificación de requisitos, elementos del sistema como interfaces de usuario, formatos de reportes se pueden extraer directamente usando prototipos, así como también otros requisitos pueden ser inferidos haciendo corridas experimentales con el prototipo (IEEE, 1998).

En este sentido, es importante destacar que la preocupación más importante en la industria del software es desarrollar productos con calidad (Pressman, 2010). A pesar de esto, existen muchos ejemplos de software de baja calidad, lo que resulta en la insatisfacción de los usuarios y en grandes pérdidas económicas (Howles, 2003). Por su parte, López y Agüero, (2007) mencionan que el rápido crecimiento de la tecnología de información resulta en un gran número de software, pero que frecuentemente los usuarios se encuentran insatisfechos con su calidad, ya que evaluar la calidad en el software puede resultar una tarea complicada, por lo general los proyectos de desarrollo de sistemas de información implementan planes de calidad durante el proyecto, esto proporciona una manera ordenada y controlada para llevar a cabo el proceso de desarrollo, pero esto no garantiza que el sistema resultante tendrá la calidad deseada.

De este modo se define calidad como el "Grado en el que un conjunto de características inherentes cumple con los requisitos" (ISO, 2000, p. 23). Donde una característica es un rasgo diferenciador y un requisito es una necesidad o expectativa establecida, generalmente implícita u obligatoria. Según Dolado y Fernández (2000) la definen como la valoración por los consumidores de la superioridad o excelencia de un bien o servicio.

Por consiguiente, Shneiderman y Plaisant (2010), argumentan que la medición de la calidad es clave para lograr desarrollar productos de alta calidad, sobre todo porque la evaluación depende, en gran parte, de la visión del usuario. El éxito de un sistema de información, más que depender de un nivel de calidad evaluado internamente, se

relaciona con el grado de aceptación y satisfacción del usuario que, no sólo se conforma con que el software cumpla con los requisitos funcionales, sino que le otorga gran importancia a la interfaz de usuario, en cuanto a su apariencia, retroalimentación y usabilidad, haciendo que el diseño de la interfaz de usuario sea un ingrediente primordial y esencial en la calidad final de la aplicación

Así mismo, Pfleeger (2008) expresa que existen diferentes modelos de calidad que indican cuáles son las cualidades deseables para determinar la calidad de un producto de software y aunque se definen criterios para medir calidad, es necesario establecer métodos de evaluación que permitan obtener resultados objetivos, integrando tanto los elementos técnicos a evaluar como la participación continua del usuario. De este manera, los usuarios asignan una buena calidad al software si este hace lo que ellos desean, en una forma que sea fácil de aprender y fácil de utilizar; ya que ellos son quienes realmente saben cuáles son sus necesidades y los resultados que esperan obtener. Existen aspectos propios del usuario tales como estilo de aprendizaje, herramientas favoritas, e incluso diferencias culturales, los cuales pueden impactar en la eficiencia del sistema en cuestión. Este aspecto humano de la Ingeniería de Software puede ser el más crítico, ya que un sistema puede tener un desempeño excelente realizando una función, pero si los usuarios no pueden entender cómo usarlo, el sistema es un fracaso (Pfleeger, 2008).

De allí la importancia que lo usuarios sean integrados de una manera formal dentro del ciclo de desarrollo de software, desde el inicio hasta el final; y de una manera intensiva en el proceso de gestión de requisitos, donde a través de la validación de requisitos usando prototipos de interfaces de usuario puedan evaluar la calidad del sistema futuro. Esta iniciativa no es nueva, se evidencia en las sesiones JAD[4] (*Joint Application Design*) y de prototipado, en donde se invirtieron grandes esfuerzos para crear un consorcio con los actores interesados en el desarrollo de sistemas, así mismo métodos de desarrollo de software recientes como XP[5] (*Extreme*

[4] JAD: Desarrollo en Conjunto de Aplicaciones, http://dl.acm.org/citation.cfm?id=213690
[5] XP: Programación Extrema, http://www.extremeprogramming.org/

Programming) y SCRUM[6], por ejemplo, tratan de agilizar los procesos de desarrollo de sistemas, involucrando a los usuarios finales desde el inicio y creando un ciclo de iteraciones para ir refinando los requisitos válidos para la construcción del software. (Berenbach, Paulish, Kazmeier y Rudorfer, 2009). Sin embargo aún existen muchas deficiencias en este sentido y la Ingeniería de Requisitos sigue siendo un punto álgido dentro de los proyectos de software (Montilva y otros 2008).

En otro orden de ideas, Lujan (2002) sostiene que con la introducción de Internet y de la Web, se abrieron infinidad de posibilidades en cuanto al acceso a la información desde casi cualquier sitio, la Web ha transformado los sistemas informáticos, ha roto las barreras físicas (debido a la distancia), económicas y lógicas (debido al empleo de distintos sistemas operativos, protocolos, entre otros). Las aplicaciones Web son cada día más comunes, debido a la popularidad y extensión que tiene la Internet, aunado con la facilidad para usar, actualizar y mantener estas aplicaciones sin distribuir e instalar software (Buzzo y Rivero, 2009).

Cuando la Web se inició, se encontraba en un entorno estático, con páginas en HTML básicas que sufrían pocas actualizaciones y tenían poca o ninguna interacción con el usuario, las aplicaciones Web que existían presentaban interfaces difíciles de usar, con demora de tiempo para pasar de una pantalla a otra, lo que hacía que la experiencia del usuario fuera poco amigable y agradable, siendo un ambiente problemático para la disponibilidad de aplicaciones que requerían interfaces robustas con el usuario (Mahemoff, 2006). Esto fue cambiando a medida que las necesidades fueron creciendo, incorporando aplicaciones Web orientadas al usuario, surgiendo lo que se denominó la Web 2.0, que es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de la Web enfocadas y centradas completamente en el usuario. Se trata de aplicaciones que generan colaboración y entornos de servicios que reemplazan las aplicaciones de escritorio. Según O'Reilly y Battelle (2009), el término Web 2.0 hace referencia a un conjunto de aplicaciones usadas en la Web con el objetivo único de darle al usuario el control de

[6] SCRUM: http://www.scrum.org/

sus datos. La idea de esto es mantener la misma funcionalidad, pero de mejor forma, más eficiente, más confiable, más confortable, es decir a grandes rasgos mejorar la interacción del usuario con la aplicación.

Así, Martínez-Ruiz, Vanderdonckt y Arteaga (2010) afirman que la tendencia actual de orientar las aplicaciones Web lo más parecidas, en cuanto a sus funcionalidades y características, a las aplicaciones de escritorio, y al desarrollo reciente de la comunicación asincrónica, han producido una nueva generación de aplicaciones Web denominadas *Rich Internet Applications*: RIA[7], las cuales están orientadas a satisfacer las expectativas del usuario en términos de usabilidad, fiabilidad, calidad, facilidad de mantenimiento y el rendimiento.

El termino RIA se refiere a una familia heterogénea de soluciones, que se caracterizan por el objetivo común de añadir nuevas capacidades a las aplicaciones Web convencionales. Estas combinan la arquitectura de distribución ligera de la Web, con la interactividad de las interfaces de las aplicaciones de escritorio, obteniendo como resultado una mejora en todos los elementos de una aplicación Web (datos, lógica de negocio, comunicación y presentación), representando un gran área de interés en el desarrollo de software en los últimos años (Rossi y otros, 2010).

Por consiguiente, surge entonces la necesidad de desarrollar aplicaciones Web enriquecidas, lo cual impulsa al surgimiento de productos de software orientados a su ejecución y desarrollo, esto se evidencia en un gran número de frameworks, proyectos y tecnologías que apuntan a proveer el desarrollo ágil de RIA, a través del uso de lenguajes específicos y a lograr eficiencia en su ejecución, facilitando además la portabilidad de las mismas a una variedad de plataformas heterogéneas como, por ejemplo, dispositivos móviles (Buzzo y Rivero, 2009). El creciente desarrollo de las RIA ha conllevado a la detección de patrones de diseño comúnmente aplicados en su construcción, surgiendo de esta forma los patrones RIA, que no son más que patrones de diseño Web. Un patrón de diseño Web es una descripción de un problema de

[7] RIA: Aplicaciones Web Enriquecidas, http://www.adobe.com/resources/business/rich_internet_apps/

diseño Web y buenas soluciones a este problema, donde el problema que viene a solucionar el patrón es un problema de interfaz de usuario, de interacción, usabilidad o prestación en la misma, para lograr mejorar tanto la funcionalidad del sitio como la amigabilidad del mismo. Así, los patrones RIA, son un vocabulario común y un conjunto de normas para diseñar las interfaces de las RIA y lograr así la interactividad, amigabilidad y usabilidad que las caracteriza (Scott, B. y Neil, 2009).

De lo anteriormente expuesto, se evidencia que el crecimiento desenfrenado de la Web está ocasionando un impacto en la sociedad y el nuevo manejo de la información en las diferentes áreas en que se presenta, se traduce en el crecimiento de las aplicaciones Web y por lo tanto en un desafío para los Ingenieros de Software (Buzzo y Rivero, 2009), de forma tal que en el área de la Ingeniería de requisitos, disciplina que está llamada a combatir los problemas que aún existen en el proceso de desarrollo de software, requiere de una revisión y evaluación constante de las herramientas, métodos, técnicas que permitan dar apoyo a esta disciplina (Montilva y otros, 2008). En el proceso de validación de requisitos, los prototipos de interfaz representan una opción viable y factible, y en el contexto de las aplicaciones Web los prototipos basados en patrones RIA, los cuales garantizan interfaces interactivas y amigables que mejoran la satisfacción del usuario, podría ser una buena técnica para lograr alcanzar los objetivos en este proceso de validación de requisitos. Es importante resaltar que los actuales paradigmas de la Web, intentan responder a las necesidades de los usuarios, los cuales cada vez son más exigentes y participativos y demandan más incursión en los productos de software que requieren para satisfacer sus necesidades.

Por lo tanto, según la problemática antes descrita, la presente investigación se conduce hacia el diseño y construcción de un método que permitirá al ingeniero de software validar requisitos de software a partir de prototipos de interfaces de usuario basado en patrones *Rich Internet Applications* (RIA), enmarcado en un modelo de calidad. De esta manera se puede mantener e incrementar la participación del usuario final, así el proceso de desarrollo de software estará orientado hacia las necesidades manifestadas continuamente por los usuarios, todo esto inmerso en un proceso

iterativo e incremental.

Sobre la base de lo antes planteado, surgen las siguientes interrogantes que contribuyen al desarrollo de esta investigación:

- ¿Cuáles son las tendencias actuales que apoyan el proceso de validación de requisitos funcionales de software apoyado en prototipos de interfaz?
- ¿Cuál es el flujo de trabajo que debe seguir un método que facilite la validación de requisitos funcionales de software apoyado en prototipos de interfaz?
- ¿Cuáles parámetros de medición de calidad se deben utilizar para la evaluación de la calidad del prototipo de interfaces de usuarios basado en patrones RIA?
- ¿Cómo se construye y valida un método de validación de requisitos funcionales de software mediante prototipos de interfaces de usuario que incorpore elementos como patrones Rich Internet Applicactions, bajo un enfoque de calidad?

Finalmente, es propio señalar que dichas interrogantes alcanzan sus respuestas a través de la consecución de los objetivos específicos expuestos en el presente estudio.

Objetivos de la Investigación

Objetivo General:

Diseñar un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones Rich Internet Applications (RIA), bajo un enfoque de calidad.

Objetivos Específicos:

1. Analizar las tendencias actuales en cuanto a los métodos, procesos y/o modelos que den apoyo al proceso de validación de requisitos funcionales de software.

2. Determinar el flujo de trabajo que debe seguir un método que facilite la validación de requisitos funcionales de software apoyados en prototipos de interfaces de usuario.

3. Definir el conjunto de métricas a utilizar por cada uno de los factores de calidad a ser aplicados en la evaluación del prototipo de interfaces de usuarios basado en patrones RIA.

4. Construir un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basadas en patrones Rich Internet Applications, bajo un enfoque de calidad.

5. Validar el método construido a través de un caso de estudio.

Justificación e Importancia

Para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado ó codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. La calidad con que se realice la captura de los requisitos va a influenciar en todo el proceso de desarrollo del software repercutiendo en el resto de las fases de desarrollo del mismo (Pressman, 2010).

Según, Aouad y Arayici (2010), a diario se cometen errores con respecto a los requisitos, en los proyectos de desarrollo de software, esto debido a la falta de información o de conocimiento acerca del tema. Entre los errores más comunes se encuentra la implicación insuficiente del cliente, ya que este no comprende la importancia de trabajar con rigor en la obtención de los requisitos para garantizar la calidad de los resultados, trayendo consigo a largo plazo problemas en la satisfacción del producto obtenido. Otro de los errores son los requisitos crecientes y cambiantes, lo cual puede incrementar o modificar funcionalidades ya implementadas, que por no quedar claros en un principio, generalmente aumentan costos y agendas planificadas, obteniéndose finalmente un producto con serias deficiencias técnicas, tal como lo demuestra Standish Group (2009) en sus estudios realizados.

Por consiguiente se evidencia la importancia del proceso de elicitación y validación de requisitos en el ciclo de vida del software ya que no es suficiente solo elicitar los requisitos sino que también es necesario establecer técnicas efectivas que garanticen que dichos requisitos sean claros y comprensibles tanto para el ingeniero de requisitos como para el usuario final. Así, es indispensable validar dichos requisitos antes de pasar a la siguiente etapa del ciclo de vida del software. En este sentido, Sommerville (2005) indica que una técnica que ayuda enormemente a validar los requisitos obtenidos con los usuarios, es la elaboración de prototipos de interfaz, que ofrezcan a los usuarios una forma rápida de tener una visión preliminar del sistema final.

Sumado a lo anteriormente expuesto, actualmente los usuarios son cada vez más exigentes con la calidad de las aplicaciones que usan y en gran medida, su opinión respecto a las mismas, está fundada casi de forma única en su experiencia con las interfaces de usuario. Así, los usuarios finales consideran que un software tiene una alta calidad, si este hace lo que ellos desean, en una forma que sea fácil de aprender y fácil de utilizar; ya que ellos son quienes realmente saben cuáles son sus necesidades y los resultados que esperan obtener. Este aspecto humano de la Ingeniería de Software puede ser el más crítico, ya que un sistema puede tener un desempeño excelente realizando una función, pero si los usuarios no pueden entender cómo usarlo, el sistema es un fracaso (Pfleeger, 2008).

De allí la importancia que lo usuarios sean integrados de una manera formal dentro del ciclo de desarrollo de software, desde el inicio hasta el final; y de una manera intensiva en el proceso de gestión de requisitos, donde a través de la validación de requisitos usando prototipos de interfaces de usuario pueden evaluar la calidad del sistema futuro.

Por otra parte, hoy en día el desarrollo del software está orientado al empleo de tecnología Web, la cual abre infinidad de posibilidades en cuanto al acceso a la información desde casi cualquier sitio, convirtiendose en parte integral de la vida diaria de cualquier individuo y en un elemento fundamental de la sociedad. De este modo, Martínez-Ruiz y otros (2010) afirman que la tendencia actual es el desarrollo de aplicaciones Web orientadas al usuario, haciéndolas lo más parecidas, en cuanto a sus funcionalidades y características, a las aplicaciones de escritorio. Este último es el objetivo principal de la Web 2.0, la cual es la representación de la evolución de las aplicaciones Web tradicionales hacia aplicaciones Web enfocadas al usuario, manteniendo la misma funcionalidad, pero de mejor forma, más eficiente, más confiable, más confortable, es decir a grandes rasgos mejorando la interacción del usuario con la aplicación Web (O'Reilly y Battelle, 2009).

De este modo, según Rossi y otros (2010) todas las características tecnológicas de la Web 2.0 se han concentrado en las RIA, las cuales están orientadas a satisfacer las

expectativas del usuario en términos de usabilidad, funcionalidad y fiabilidad, para tratar de garantizar así, la satisfacción del usuario.

Por todo expuesto, la principal importancia y justificación de esta investigación se basa en proponer un método de validación de requisitos funcionales de software para aplicaciones Web. Esta validación se realiza a partir de un prototipo de interfaces de usuario basados en patrones RIA, los cuales garantizarán interfaces interactivas y amigables que mejorarán la satisfacción del usuario. Todo esto bajo un enfoque de calidad el cual consiste en que el usuario evalué la calidad del prototipo basado en patrones RIA, a partir de factores de calidad del modelo ISO/IEC 25010, lo cual puede ayudar a lograr alcanzar los objetivos en este proceso de validación.

Por consiguiente, se espera que el resultado de la investigación sea de importancia en el área de la Ingeniería de Requisitos, pudiendo obtener los siguientes beneficios:

- Ayudar a validar los requisitos del usuario.
- Establecer una mejor concordancia entre el sistema y las necesidades del usuario, permitiendo la posibilidad de desarrollar un sistema que satisfaga en mejor forma las necesidades y expectativas de los usuarios.
- Ayudar a garantizar el éxito de la Ingeniería de Requisitos, permitiendo validar eficazmente los requisitos obtenidos y obtener nuevas necesidades que surjan en la revisión del prototipo.

Con relación al aporte a la industria de desarrollo de software basada en aplicaciones Web, el método propuesto puede ser incorporado en las organizaciones, dentro del método de desarrollo de software que estén utilizando, para así ayudar a garantizar la calidad del producto resultante y por lo tanto la satisfacción del usuario.

Finalmente, se espera que el presente estudio sea de importancia a nivel académico, siendo utilizado como referencia para realizar otros estudios que surjan a partir de la problemática aquí especificada y para impulsar nuevos proyectos en la comunidad científica.

Alcance de la Investigación

El alcance del presente estudio está orientado al diseño y construcción de un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad.

El método a proponer se basa en la validación de requisitos funcionales a partir de prototipos construidos con cualquier herramienta que exista en el mercado, con soporte para generar prototipos para aplicaciones Web de forma automática, y que permita incorporar los patrones RIA. Aunque la investigación se centra en la validación de requisitos funcionales, también se destaca la importancia de los requisitos no funcionales, tal como lo es la usabilidad, debido a que estos son de gran importancia en un proceso de validación de requisitos funcionales mediante prototipos de interfaces de usuario.

De esta forma, a partir del prototipo de interfaces de usuarios generado, y con el apoyo del documento de especificación de requisitos, diagramas de casos de uso, diagrama de actividades, diagramas de clases y de objetos, se validan los requisitos funcionales con los usuarios finales. Para el proceso de validación se deben realizar sesiones entre el Ingeniero de Software y diferentes grupos de usuarios, donde cada grupo de usuario debe ser seleccionado según los roles que cumplan dentro del sistema, identificados en los casos de uso.

Así, las sesiones de validación están basadas en escenarios, donde cada grupo de usuarios previamente definido según su rol, valida cada uno de los escenarios, definidos en los diagramas de casos de uso, usando el prototipo de interfaces de usuario basado en patrones RIA generado. De este modo el usuario puede comenzar a experimentar la interacción con un elemento dinámico que le permita ayudar a validar si realmente el prototipo cumple con sus necesidades además de ir refinando sus exigencias y expectativas para el sistema que está naciendo.

Una vez que el usuario haya interactuado con el prototipo validando los diferentes escenarios según su rol o roles definidos, procede a evaluar la calidad del prototipo

por parte del usuario, a partir de un instrumento que incorpore algunos factores de calidad del modelo ISO/IEC 25010. Los factores de calidad a usar serán asociados con los patrones RIA que tengan relación, para así evaluar la calidad de una manera objetiva, clara y precisa. Los factores a seleccionar para evaluar la calidad del prototipo de interfaces de usuario basado en patrones RIA a utilizar, fueron tomados a partir de un estudio realizado por Macías y Gómez (2010), el cual es explicado en la sección Antecedentes, para luego ser adaptados al modelo de calidad ISO/IEC 25010. Para finalizar, una vez diseñado el método propuesto se ejemplifica su uso, a través de un caso de estudio.

Es importante destacar que el método de investigación a seguir para llevar a cabo el presente estudio, se basa en el método para la investigación en Ingeniería del Software propuesto por Marcos y Marcos (1998), el cual está compuesto por seis (6) etapas, que se describen con detalle en la sección Bases Teóricas. De las etapas que conforman dicho método, la presente investigación llega hasta la cuarta etapa denominada Resolución, Validación y Verificación, quedando el resto de las etapas para trabajos futuros.

CAPÍTULO II

MARCO TEÓRICO

El presente marco teórico sustentará la investigación para la realización del diseño de un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad. Este capítulo está conformado en dos (2) secciones las cuales se describen a continuación:

Antecedentes de la Investigación

La técnica de crear prototipos de interfaces de usuario, es un método muy utilizado en la actualidad como una herramienta efectiva para la validación de requisitos funcionales de software, dentro de la disciplina de la Ingeniería de Requisitos, esto aunado a la importancia que los usuarios finales le otorgan a elementos como la presentación, funcionalidad, interactividad, y usabilidad, conforman un conglomerado de características que es común encontrar en las interfaces de usuario de aplicaciones Web a través del uso de patrones RIA. Es por ello que en esta sección se analizan los aportes obtenidos a través de algunos trabajos de investigación vinculados al tema tratado, lo cual permitirá obtener lo respaldos necesarios para dar estructura firme al estudio y servirán de base para dirigir la investigación en aras de obtener la solución del problema planteado. Estos trabajos se describen brevemente a continuación:

Validación de Requisitos de Software a través de Prototipos de Interfaces de Usuario

En el 2008, Montilva y otros, en su trabajo presentado denominado "*Gray Watch, Método de Desarrollo de Software para Aplicaciones Empresariales*", propusieron un marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial. Dentro de este método de desarrollo de software, existe una fase destinada a las actividades pertenecientes a la Ingeniería de Requisitos, la cual consiste en determinar y documentar los requisitos funcionales y no-funcionales que los actores del Sistema de Negocios tienen con respecto a la aplicación empresarial que se va desarrollar. Allí se plantea que la Ingeniería de Requisitos requiere de la ejecución de cinco (5) subprocesos complementarios para especificar los requisitos de una aplicación empresarial, como lo muestra la figura 1. De estos subprocesos, es importante destacar para el presente estudio, el subproceso de validación de requisitos, el cual Montilva y otros, definen como el proceso de evaluación y revisión sistemática del documento de requisitos para asegurar que éste define a la aplicación de software correctamente.

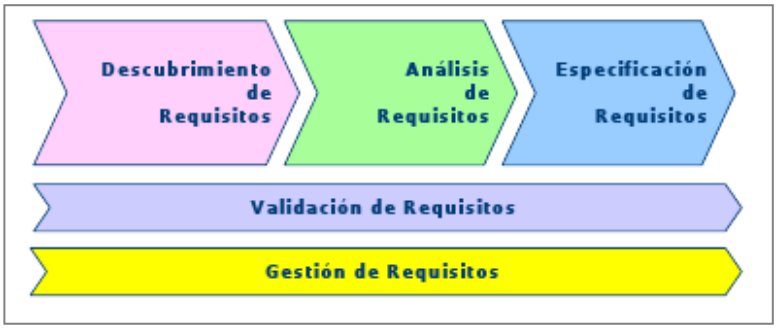

Figura 1. Proceso de Ingeniería de Requisitos.
Fuente: Montilva y otros (2008)

El flujo de trabajo del subproceso de validación de requisitos se muestra en la figura 2. En la tabla 2 se describen detalladamente las actividades del subproceso.

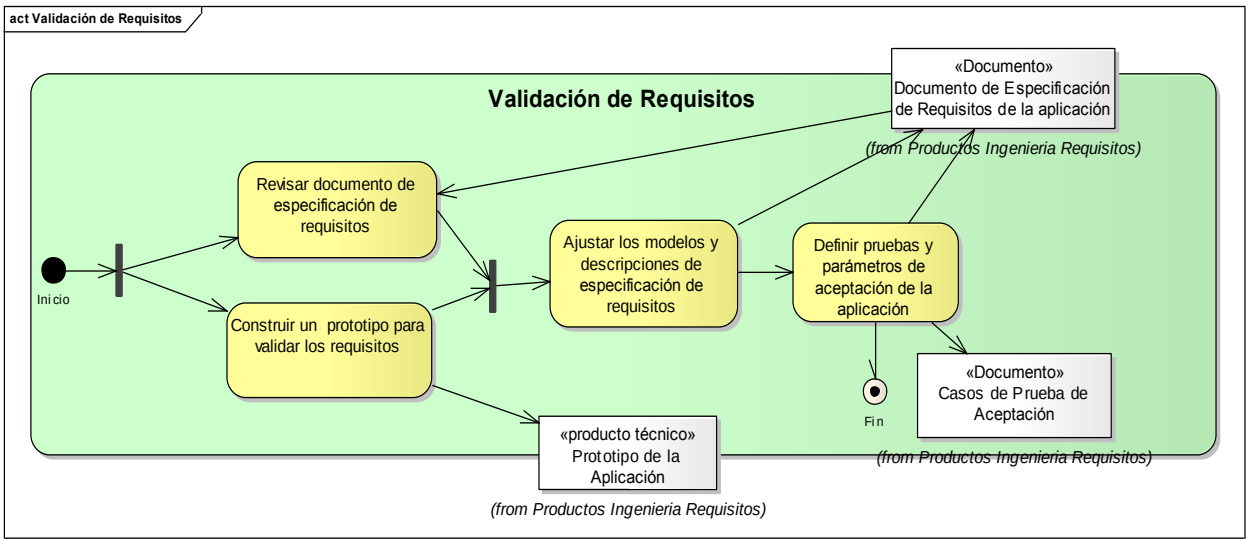

Figura 2. Flujo de trabajo del Subproceso de Validación de Requisitos.
Fuente: Montilva y otros (2008)

Tabla 2. Descripción de Flujo de Trabajo: Validación de Requisitos

Actividad	Tareas	Productos
Revisar documento de especificación de requisitos	• Validar la estructura y el contenido del documento • Validar especificación técnica de los requisitos	• Documento validado.
Construir un prototipo para validar los requisitos	• Desarrollar un prototipo que emule la funcionalidad (según los casos de uso) y la interfaz que tendría la aplicación • Validar funcionalidad e interfaz de la aplicación.	• Prototipo de la aplicación.
Ajustar los modelos y descripciones de la especificación de requisitos	• Modificar los modelos y descripciones de especificación técnica atendiendo a los resultados de validación del prototipo. • Verificar consistencia e integridad de la especificación técnica.	• Modelos actualizados y validados.
Definir pruebas y parámetros de aceptación de la aplicación	• Determinar parámetros de aceptación de la aplicación. • Definir casos de prueba de aceptación. • Verificar, con los interesados, los parámetros de aceptación y los casos de prueba de aceptación de la aplicación.	• Conjunto de casos de prueba de aceptación de la aplicación.

Fuente: Montilva y otros (2008)

De este modo, el aporte principal que presenta el método Gray Watch a esta investigación, es el subproceso de validación de requisitos, sus de actividades y tareas a realizar, así como los productos (artefactos) que se usan y generan en cada actividad, los cuales servirán como base para el método a proponer en el presente trabajo, el cual tomará el flujo de trabajo definido para este subproceso, adicionándole los cambios que sean necesarios para lograr alcanzar los objetivos propuestos. A continuación se muestra un cuadro resumen de la investigación realizada por Montilva y otros (2008), (ver Tabla 3).

Tabla 3. Resumen de Montilva y otros (2008)

Objetivos	Método/ Metodología	Contribuciones	Aporte al Presente Trabajo
Definir un método de desarrollo de software bien definido y documentado, para producir una aplicación empresarial	Ingeniería de Software, Ingeniería de Métodos, Ingeniería de Requisitos, Gestión de Proyectos, Modelado de Negocios	El método Gray Watch define un método para el desarrollo de software empresarial, que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial.	El flujo de trabajo (actividades, tareas y productos), del subproceso de validación de requisitos perteneciente al proceso de Ingeniería de Requisitos

Fuente: La Autora (2012)

En este orden de ideas, se puede citar a Ogata y Matsuura (2010), quienes en su trabajo de investigación titulado "*Evaluation of a Use-Case-Driven Requirements Analysis Tool Employing Web UI Prototype Generation*", proponen un método para el análisis y validación de requisitos, el cual consiste en generar automáticamente un prototipo de interfaz de usuario a partir de un modelo de análisis de requisitos en UML para el desarrollo de aplicaciones Web empresariales.

Estos autores afirman que el análisis de requisitos dirigido por modelos de casos de uso, el cual consiste en un diagrama de casos de uso en UML y varias plantillas donde se detalla y describe cada caso de uso en lenguaje natural, es una método típico usado para el análisis orientado a objetos que especifica cómo un sistema debe interactuar con los usuarios y con otros sistemas externos, el cual en la mayoría de los casos no es suficiente para que el proceso de gestión de requisitos tenga éxito. Esto se debe a que el modelo de casos de uso define con precisión los requisitos de los clientes, puede ser manejado como especificación de requisitos, pero presenta entre otros los siguientes problemas, los términos técnicos no son entendidos por los usuarios finales, la descripción en lenguaje natural de los casos de uso hace que los desarrolladores y los usuarios se imaginen la implementación del sistema de manera arbitraria, lo que puede causar una mala comprensión de los requisitos entre los desarrolladores y los usuarios finales.

Por lo tanto, en su trabajo Ogata y Matsuura proponen un método para generar automáticamente un prototipo de interfaces de usuario a partir de un modelo de análisis de requisitos en UML (modelo RA) para el desarrollo de aplicaciones empresariales Web, de tal forma que los desarrolladores pueden resolver los problemas antes mencionados. El modelo de RA está compuesto por: diagramas de actividad, un diagrama de clases, los diagramas de objetos, escenarios y modelo de navegación. El prototipo interfaz de usuario es una herramienta para comprender fácilmente el sistema futuro e implementa aspectos específicos de un sistema sólo como interfaces de usuario.

De este modo, la idea del método propuesto por Ogata y Matsuura, es que los

desarrolladores definan un modelo RA basado en los requisitos funcionales definidos en el modelo de casos de uso; luego un prototipo de interfaces de usuario Web es generado automáticamente a partir del modelo RA definido. Una vez que el prototipo es generado, los desarrolladores y los usuarios finales realizan el proceso de validación de los requisitos a través del prototipo de interfaz de usuario Web. El resultado de este proceso son posibles problemas detectados los cuales incluyen errores de definición en los requisitos elicitados, y/o la incorporación de los requisitos descubiertos en este proceso de verificación.

De este modo, si lo amerita, se realiza el refinamiento del modelo RA, donde los desarrolladores modifican el modelo RA mediante la corrección de los problemas encontrados, y/o incorporando nuevos requisitos descubiertos en el proceso de validación de requisitos, la salida de este proceso es el modelo RA refinado. Los desarrolladores realizan nuevamente estos pasos (generación de prototipos a partir del modelo RA, el proceso de validación de requisitos, y el proceso de refinamiento del modelo RA), iterativamente hasta que los clientes estén satisfechos con el prototipo de interfaz de usuario generado lo que implica el modelo RA bien definido. La siguiente figura muestra esta secuencia de actividades correspondientes a dicho método.

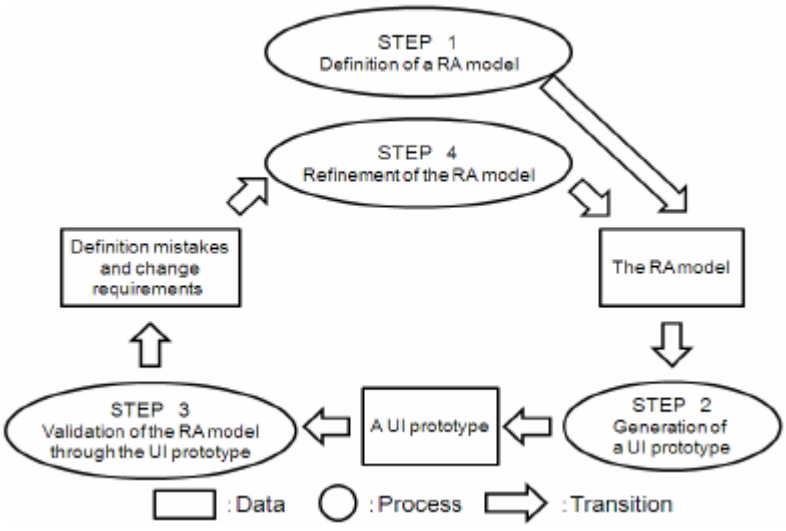

Figura 3. Actividades que conforman el método propuesto por Ogata y Matsuura. Fuente: Ogata y Matsuura (2010)

Finalmente Ogata y Matsuura hicieron una evaluación de su método a través de un caso de estudio con un grupo de usuarios, para así evaluar su efectividad y comparar su propuesta con el método tradicional de análisis de requisitos dirigido por casos de uso, donde los resultados arrojados fueron satisfactorios.

Por consiguiente, el trabajo presentado por estos autores, sirve como aporte a la presente investigación donde se tomara como base su método propuesto de analizar y validar requisitos a través del uso de prototipo de interfaces de usuario para aplicaciones Web, generado en forma automática partiendo de un conjunto de diagramas en UML, en un proceso iterativo. A continuación se muestra una tabla que presenta las características más relevantes de la investigación realizada por por Ogata y Matsuura (2010).

Tabla 4. Resumen de Ogata y Matsuura (2010)

Objetivos	Método/ Metodología	Contribuciones	Aporte al Presente Trabajo
Proponer un método para generar automáticamente un prototipo de interfaz de usuario para aplicaciones Web a partir de un modelo de análisis de requisitos en UML, con el fin de disminuir la inconsistencia y ambigüedad de los requisitos elicitados.	Unified Modeling Language (UML), Ingeniería de Requisitos, Aplicaciones Web	• Un método que permite validar requisitos a través de un prototipo generado de forma automática a partir de diagramas en UML (Modelo RA). • Disminuir el tiempo empleado para el análisis de requisitos, garantizando su efectividad en comparación con el modelo tradicional de análisis de requisitos dirigido por casos de uso.	El uso de un prototipo de interfaces de usuario para aplicaciones Web, para validar requisitos, donde el prototipo es generado automáticamente partiendo de diagramas de actividades, diagrama de clase y diagramas de objetos en UML.

Fuente: La Autora (2012)

Otra iniciativa que servirá de antecedente a este trabajo de investigación es la desarrollada por Gabrysiak, Giese y Seibel, (2011), en su trabajo de investigación titulado "*Towards Next Generation Design Thinking: Scenario-Based Prototyping for Designing Complex Software Systems with Multiple Users*", donde destacaron los beneficios de la utilización de prototipos para comunicar, validar y explorar puntos de vista e ideas de diseño de los usuarios en el proceso de ingeniería de requisitos, en un proceso iterativo e incremental.

Para que este proceso tenga éxito, es crucial entender las necesidades reales de todos los usuarios que intervienen en el proceso, con sus diferentes antecedentes y experiencias, es por ello que se deben capturar las ideas, opiniones y necesidades de los diferentes usuarios finales y su dominio. Una vez que se hayan levantado o descubierto los requisitos, es imprescindible validar estos requisitos desde los diferentes puntos de vistas, ya que cada usuario final entiende el problema y las necesidades individuales de manera diferente, ver figura 4.

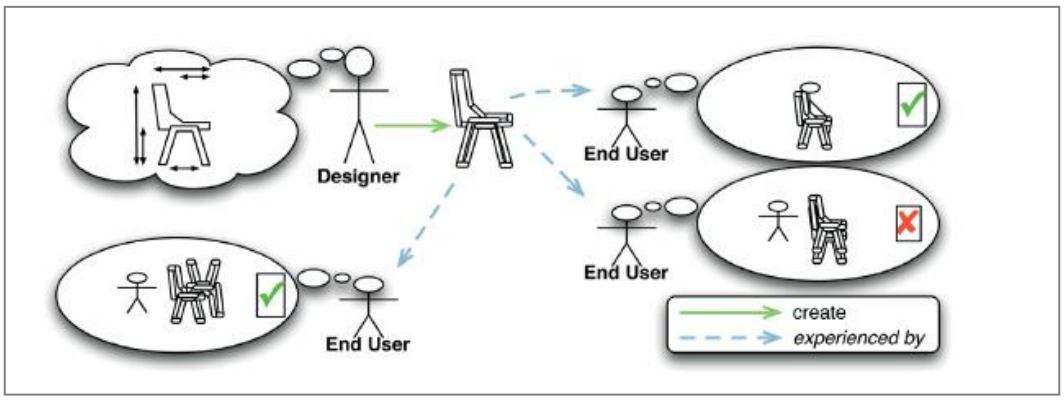

Figura 4. Cada usuario tiene su propio punto de vista.
Fuente: Gabrysiak, y otros (2011)

De este modo, Gabrysiak y otros, proponen un enfoque pensado en sistemas de software multi-usuario, donde no es suficiente usar prototipos de interfaces gráficas que representen el punto de vista de un único usuario del sistema sino que se hace necesario diseñar prototipos que impliquen a todos los usuarios ya que sólo una visión integral de todas las perspectivas entrelazadas puede garantizar que el proceso de ingeniería de requisitos tenga éxito.

Cuando los sistemas multiusuario son el centro de atención, muchos escenarios se pueden obtener durante el proceso de diseño basado en el pensamiento, a través de los modelos de Ingeniería de software (diagramas de casos de uso y actividades). Estos escenarios pueden ser considerados como instancias de proceso que necesitan que se relacionen entre sí y sean analizadas. Por ejemplo, es crucial para encontrar inconsistencias entre escenarios o para validarlos; seguir paso a paso los respectivos procesos. La modelización de escenarios y procesos, así como las técnicas de simulación son conceptos importantes para el estudio de las relaciones entre los escenarios. La información sobre los artefactos involucrados en el proceso también puede ser extraída de estos escenarios.

Por consiguiente, la visión de Gabrysiak y otros, es combinar ambos enfoques, modelos de ingeniería de software y la creación de prototipos exploratorios, donde a medida que se vayan descubriendo errores u omisiones de requisitos en las sesiones de validación mediante prototipos basadas en escenarios, se vayan incorporando las modificaciones de manera inmediata, de modo que esos errores ya estén corregidos para la siguiente sesión de validación, creando así una metodología para la captura y validación de requisitos a través de escenarios, usando prototipos, en un proceso iterativo e incremental.

La figura 5 muestra la secuencia de actividades de esta metodología, que va desde las entrevistas iniciales a la posterior validación a través de prototipos generados a partir de los modelos de ingeniería de software basado en escenarios. Inicialmente, los usuarios finales son entrevistados por los ingenieros de requisitos para recoger opiniones y necesidades Fase (1) de la figura 5. En base a estas entrevistas, se identifican las actividades, se establece un preliminar orden causal de esas actividades, se asignan los diferentes roles para cada una de esas actividades, y se definen los escenarios con sus interacciones. Estos resultados se sintetizan en los modelos de ingeniería de software Fase (2) de la figura 5, tales como diagramas de actividades, diagramas de clases para especificar conceptos de dominio, y diagramas de secuencias. Cada escenario puede ser considerado como un conjunto de diagramas de secuencia

donde se relaciona cada interacción con una especificación de comportamiento que define la condición de la interacción. Estos escenarios basados en los modelos formales son necesarios para hacer frente a la complejidad inherente de los sistemas de software multi-usuario.

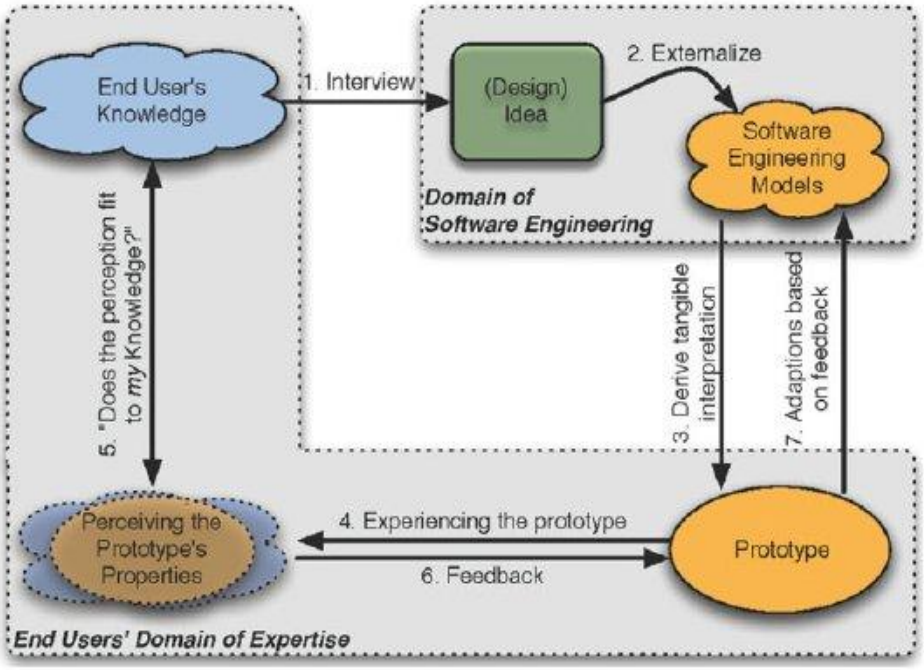

Figura 5. Metodología de recopilación y validación de ideas y necesidades de los usuarios finales.
Fuente: Gabrysiak y otros (2011)

Para validar la visión y las necesidades de los usuarios finales los autores utilizan una herramienta de generación semi-automática de un prototipo, a partir de los modelos de ingeniería de software, fase (3) de la figura 5, el prototipo es una interpretación tangible de dichos modelos. Estos prototipos están más cerca del dominio de la experiencia de los usuarios finales y, por tanto, puede ser experimentado por los ellos directamente, fase (4) de la figura 5. Durante la interacción con el prototipo, los usuarios finales perciben conceptos incorporados en el prototipo y los comparan con la comprensión correspondiente de su dominio, fase (5) de la figura 5. Esto les permite ofrecer adecuadamente una retroalimentación sobre conceptos dentro de su ámbito de competencia, fase (6) de la figura 5. Basándose en la

retroalimentación proporcionada durante la interacción con el prototipo, los modelos de ingeniería de software deben ser adaptados, fase (7) de la figura 5. Esta secuencia puede repetirse sistemáticamente hasta que exista un común entendimiento del dominio del problema.

De este modo, este artículo demuestra la factibilidad del proceso de validación de requisitos basado en escenarios, a través del uso de modelos de requisitos formales en conjunto con el uso de prototipos de interfaces de usuarios, lo cual sirve como aporte a la presente investigación donde se tomara como base la idea presentada por Gabrysiak y otros, de planificar diferentes sesiones de validación por cada grupo de usuarios clasificados en roles, basadas en escenarios, en un proceso de validación de requisitos iterativo e incremental. A continuación se muestra una tabla que resume aspectos importantes de la investigación realizada por Gabrysiak y otros (2011)

Tabla 5. Resumen de Gabrysiak y otros (2011)

Objetivos	Método/ Metodología	Contribuciones	Aporte al Presente Trabajo
Proponer una nueva visión combinando dos (2) enfoques, el uso de modelos de formales de ingeniería de software y prototipos, permitiendo la visualización interactiva para la obtención y validación de los requisitos para sistemas de software multi-usuarios.	Unified Modeling Language (UML), Ingeniería de Requisitos	Una metodología que permite que los requisitos puedan ser validados mediante la combinación de modelos de requisitos con el uso de prototipos, en sesiones de validación basadas en escenarios y grupos de usuarios (roles), de forma iterativa e incremental.	La metodología de recopilación y validación requisitos basada en escenarios que permita a cada usuario final, según su rol, realizar la validación de requisitos usando prototipos de interfaces.

Fuente: La Autora (2012)

Calidad del Software

En el 2010, Macías y Gómez, en su investigación titulada "Utilizando el Modelo de Calidad de McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas de Información por los Usuarios", realizaron un estudio en donde involucraron a un grupo de usuarios en la evaluación de la calidad de software. Primeramente ellos destacan que la finalidad de todo software es satisfacer las necesidades del usuario, el uso de aplicaciones de software dentro de las organizaciones se ha convertido en una necesidad para el buen funcionamiento y desarrollo de las mismas. Frecuentemente estas organizaciones se enfrentan a la liberación de software que no cumplen totalmente con sus necesidades, lo cual puede ir desde el incumplimiento de los requisitos establecidos hasta la facilidad de uso.

Debido a esto surge la necesidad de que el software pase por una evaluación detallada con el fin de mejorar la calidad del producto final y satisfacer las necesidades del usuario, es por ello que el proceso de medición de la calidad del software debe implementarse en todo el ciclo de vida del mismo.

Macías y Gómez, afirman que existen diferentes modelos de calidad que indican las cualidades deseables para determinar la calidad de un producto de software; sin embargo, tales modelos no establecen mecanismos definidos para su evaluación. Es por ello que su investigación propuso integrar y definir, en base a estos modelos, los principales factores que se pueden analizar y la forma de evaluarlos, considerando principalmente la integración de los usuarios finales en dicha evaluación, todo esto para contribuir a lograr aplicaciones de software de alta calidad.

De esta forma, Macías y Gómez, desarrollaron un método que consistió en establecer, en base a los modelos y estándares de calidad analizados, ¿cuáles son los factores de calidad de un producto de software que deben evaluarse desde el punto de vista del usuario?, ¿cómo deben medirse? y ¿en qué términos?, es así como proponen un conjunto de factores de calidad combinando el modelo McCall con un estándar de la norma ISO 9126-1. La selección de estos modelos se basó en el hecho de que otras alternativas, se centran en la evaluación de la calidad del proceso más no la del

producto, con lo cual la calidad en el proceso no garantiza la calidad en el producto, solo estandariza la forma de realizar las actividades.

Así, para el método de evaluación de calidad de sistemas de información por los usuarios, propuesto por Macías y Gómez, se analizaron los modelos de calidad de McCall e ISO 9126-1, mediante el cruce e integración de sus atributos, identificando cuáles de ellos podían evaluarse desde el punto de vista del usuario final (Ver figura 6 y la Tabla 6). Dichos modelos fueron seleccionados ya que concuerdan en muchos de los factores evaluables, existen otros factores en los modelos integrados pero son evaluables desde el punto de vista del desarrollador, por lo cual no se incluyeron en dicha propuesta.

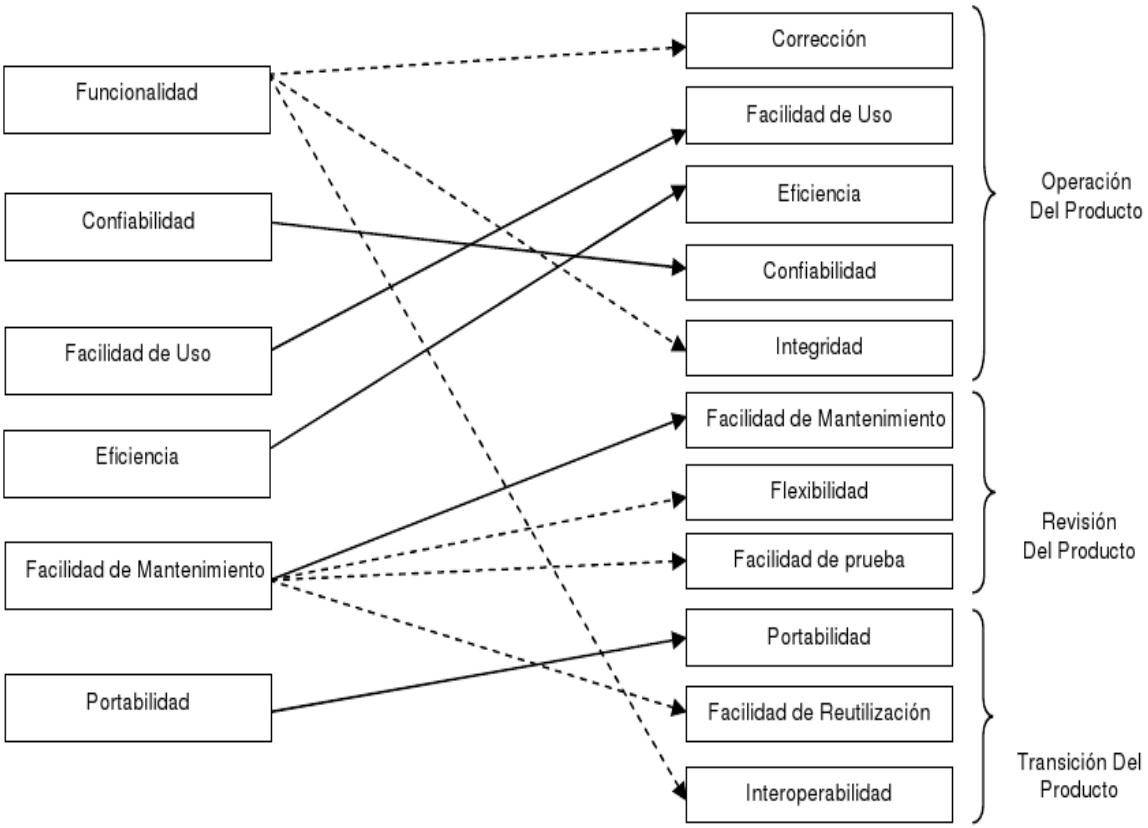

Figura 6. Equivalencias McCall - ISO 9126
Fuente: Macías y Gómez (2010)

Tabla 6. Factores para Evaluar la Calidad del Software por parte del Usuario

Características de Calidad	Factor	Descripción
Funcionalidad	Conformidad	El software tiene capacidad de proporcionar funciones adecuadas.
	Exactitud	El software proporciona resultados correctos, precisos y satisface los objetivos del cliente.
	Seguridad	El software cuenta con mecanismos de seguridad y control de accesos no autorizado a los datos.
Confiabilidad	Madurez	El software tiene la capacidad de evitar errores
	Tolerancia a Fallas	El software está preparado para mantener un desempeño adecuado en caso de fallas.
	Recuperabilidad	El software que considera las acciones necesarias para recuperarse en caso de fallas de manera fácil y rápida.
Facilidad de Uso	Comprensibilidad	El software es fácil de entender en su conveniencia y en cómo puede ser utilizado.
	Facilidad de Aprendizaje	El software permite al usuario aprender su uso de manera fácil y rápida.
	Operabilidad	El software facilita al usuario su operación y control.
Eficiencia	Tiempo de Respuesta	El software cuenta con tiempos adecuados de respuestas al realizar sus diferentes funciones.
Portabilidad	Adaptabilidad	El software puede adaptarse a diferentes entornos sin necesidad de aplicar acciones específicas diferentes a las establecidas.

Fuente: Macías y Gómez (2010)

Entonces, una vez definidos los factores para evaluar la calidad del software por parte de los usuarios, Macías y Gómez crearon un instrumento de evaluación que consistió en un cuestionario conformado por una serie de preguntas así como la escala de respuestas posibles. El siguiente paso fue evaluar su propuesta, a través de un caso de estudio en donde involucraron a los usuarios en la evaluación de la calidad de software, donde participaron dos grupos de usuarios diferentes evaluando dos prototipos del mismo sistema. De acuerdo a los resultados obtenidos, se pudo

comprobar que al integrar al usuario en la evaluación de software, ayuda a incrementar su calidad significativamente, indicando que los usuarios ponen especial énfasis en ciertos aspectos de los sistemas de información que en ocasiones los desarrolladores tienden a omitir.

Por consiguiente, la investigación de Macías y Gómez (2010), tiene aportes en el ámbito del presente trabajo demostrando la importancia de la evaluación de la calidad de los sistemas de información por parte de los usuarios, así como también los factores de calidad propuestos para dicha evaluación, que fue el resultado de la integración de los modelos McCall e ISO 9126-1. Estos factores de calidad presentados en el método de Macías y Gómez, serán usados en el presente estudio, para incorporarlos en el método de validación de requisitos a partir de prototipos de interfaces de usuario basado en patrones RIA a presentar, así tomando como base estos factores se evaluará la calidad del prototipo por parte de los usuarios, lo cual ayudará en el proceso de validación de requisitos. A continuación se muestra un cuadro resumen de la investigación de Macías y Gómez (2010), (ver Tabla 7).

Tabla 7. Resumen de Macías y Gómez (2010)

Objetivos	Método/ Metodología	Contribuciones	Aporte al Presente Trabajo
Aplicar un método para la evaluación de un producto de software donde los usuarios finales participen, tomando como base factores de calidad de modelos existentes.	Calidad de Software, Ingeniería de Software.	Definición de los principales factores de calidad, producto de combinar los modelos de calidad McCall e ISO 9126-1, usados para evaluar la calidad de un sistema de información por parte de los usuarios.	La importancia de la evaluación del software por parte de los usuarios. Los factores de calidad usados para la evaluación de software por parte de los usuarios.

Fuente: La Autora (2012)

Rich Internet Applications

En el 2009, Scott y Neil publicaron un libro técnico especializado titulado "*Designing Web Interfaces. Principles and Patterns for Rich Interaction*", el cual trata sobre el diseño de una interacción enriquecida en la Web, presentaron una selección de las mejores prácticas, modelos y principios que se deben incorporar en las interfaces que van a conformar una aplicación Web enriquecida, para crear una experiencia única en el usuario. Los autores fundaron un equipo de diseño para mejorar la experiencia del usuario, y trabajaron para mejorar decenas de productos, realizando evaluaciones heurísticas y participando en el rediseño completo de aplicaciones Web. A partir de ese trabajo, se logró obtener una serie de patrones de diseño de interfaces de usuario, así como los anti-patrones que representan los errores más comunes a evitar. De esta forma, los patrones presentados en dicho libro ya fueron probados e incorporados en aplicaciones Web enriquecidas. Scott trabajo en la empresa Yahoo[8] donde publicó la librería de patrones de diseño de Yahoo[9], del mismo modo, Neil es una diseñadora de interfaces, la cual ha perfeccionado un conjunto inicial de principios y patrones de diseño, mientras dirigía más de treinta (30) aplicaciones Web enriquecidas así como sitios Web de cara al público. Estos patrones le han permitido, a ella y a sus clientes, obtener un vocabulario común y un conjunto de normas, las cuales representan un estándar, para trabajar con el diseño de nuevas aplicaciones y rediseñar aplicaciones existentes. Por lo tanto, más de treinta (30) años de experiencia de estos autores fue recopilada en este libro para presentar un conjunto de principios y patrones para el diseño de interfaces Web enriquecidas.

Scott y Neil, proponen seis (6) principios de diseño simples para agrupar los patrones y las mejores prácticas para la creación de aplicaciones de internet enriquecidas, los cuales se explicarán en la siguiente sección (Bases Teóricas).

---

[8] Yahoo: es una empresa prestadora de múltiples servicios en Internet una de las más populares del mundo, http://www.yahoo.com/

[9] http://developer.yahoo.com/ypatterns/

De este modo, la selección de los patrones para el diseño de interfaces Web enriquecidas, realizada por Scott y Neil, la cual fue agrupada y categorizada en seis grandes principios, es de importancia para la presente investigación ya que parte de estos elementos deben estar incorporados en el prototipo de interfaces de usuarios a evaluar en el método de validación de requisitos propuesto. En otras palabras, el método a desarrollar se basará en sesiones de validación de requisitos con los usuarios a partir de un prototipo de interfaces, el cual debe incorporar algunos de los patrones RIA mencionados por estos autores. En síntesis, un cuadro de los principales aspectos del libro de Scott y Neil (2009), relacionados con la presente investigación, se muestra a continuación (ver tabla 8).

Tabla 8. Resumen de Scott y Neil (2009)

Objetivos	Método/ Metodología	Contribuciones	Aporte al Presente Trabajo
Recopilar más de trienta años de experiencia de los autores en el diseño de interfaces, en libreo que presenta un conjunto de principios y patrones para el diseño de interfaces Web enriquecidas.	Rich Internet Applications, Tecnología, Patrones RIA, Diseño pensado en el Usuario.	Presentar una selección de las mejores prácticas, modelos y patrones, agrupados en seis grandes principios. Estos patrones deben guiar el diseño de las interfaces que van a conformar una aplicación Web en. riquecida, para crear una experiencia única en el usuario.	Los patrones presentados deben estar incorporados en algún grado en el prototipo de interfaces de usuarios a evaluar en el método de validación de requisitos propuesto.

Fuente: La Autora (2012)

Finalmente, los cinco (5) antecedentes presentados, señalaron la importancia y la vigencia que tiene el tema en estudio, donde cada uno de ellos aportan ciertas características que sirven de base para la solución del problema planteado.

Bases Teóricas

En esta sección se exponen los fundamentos teóricos, conceptos y definiciones necesarios para entender el cuerpo de conocimiento en el que se basa la propuesta que se presenta en este trabajo de investigación, los cuales están estructurados tal como se muestra en la figura 7:

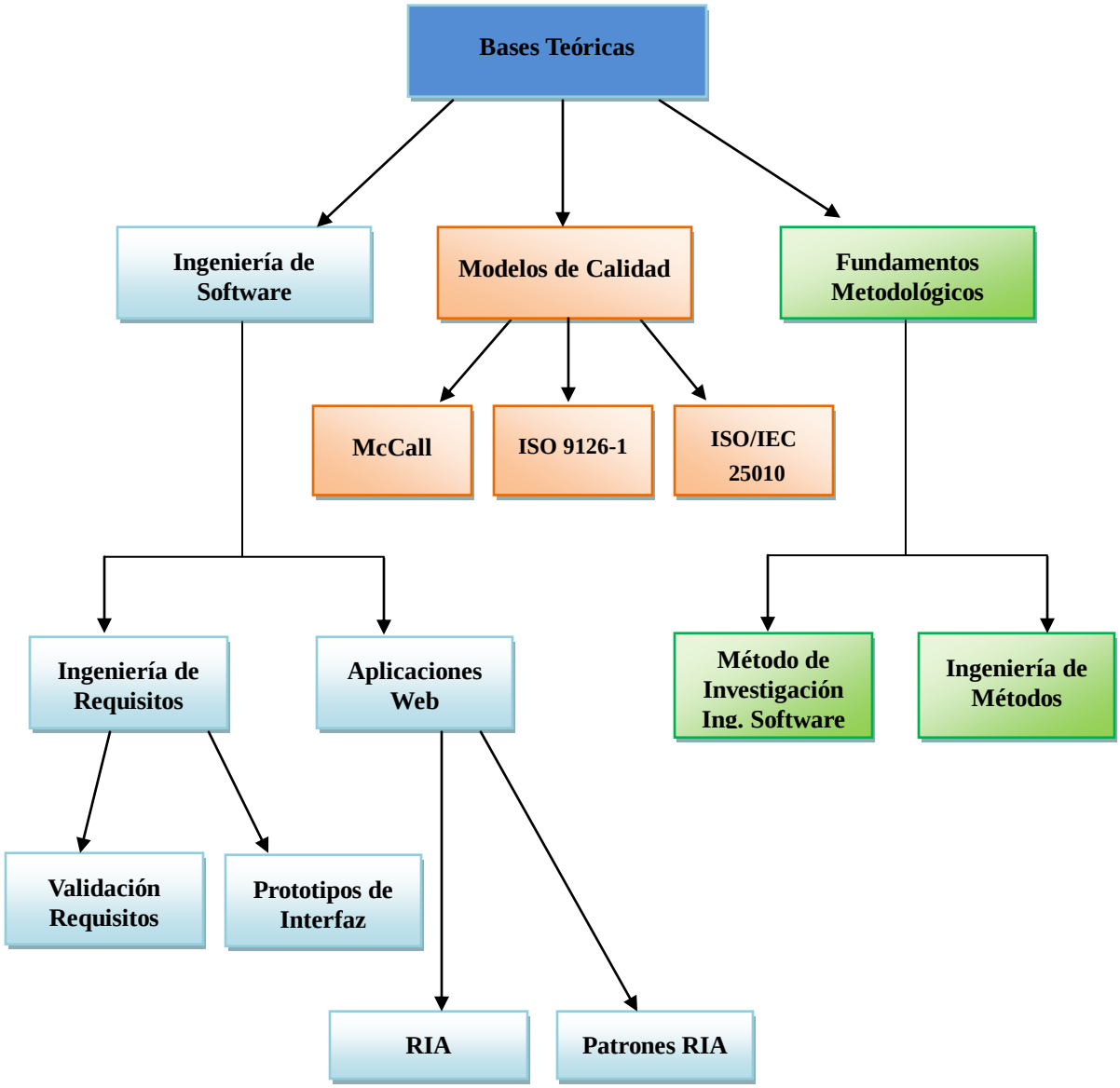

Figura 7. Mapa Conceptual Bases Teóricas
Fuente: La Autora (2012)

Ingeniería de Software

El término Ingeniería de Software empezó a usarse a finales de la década de los sesenta, para expresar el área de conocimiento que se estaba desarrollando en torno a las problemáticas que ofrecía el software en ese momento.

Sommerville (2005), sostiene que en aquella época el crecimiento espectacular de la demanda de sistemas de computación cada vez más y más complejos, asociado a la inmadurez del propio sector informático y a la falta de métodos y recursos, provocó lo que se llamó la crisis del software, donde los grandes proyectos tenían años de retraso, costaban mucho más de lo presupuestado, eran irrealizables, difíciles de mantener y con un desempeño pobre. Se necesitaban nuevas técnicas y métodos para controlar la complejidad inherente a los grandes sistemas.

Se han hecho enormes progresos desde 1968 hasta el presente con lo cual el creciente desarrollo de la ingeniería de software ha mejorado considerablemente los sistemas producidos. Se comprenden mucho mejor las actividades involucradas en el desarrollo de software y se han desarrollado métodos efectivos de especificación, diseño e implementación de software; las notaciones y herramientas existentes reducen el esfuerzo requerido para producir sistemas grandes y complejos.

Pressman (2010), define Ingeniería del Software como "la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software". (p. 12).

En este orden de ideas, Sommerville (2005) define Ingeniería del Software como:

> La Ingeniería de Software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza, existiendo dos frases claves:
>
> 1.- Disciplina de la Ingeniería. Los ingenieros aplican teorías, métodos y herramientas donde sean convenientes, utilizándolas de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlos.

> 2.- Todos los aspectos de la producción de software. La ingeniería de software no solo comprende los procesos técnicos del desarrollo, sino también las actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías que apoyen a la producción de software.(p. 6).

En tal sentido, una de las disciplinas principales de la Ingeniería de Software que es de gran importancia en la presente investigación es la Ingeniería de Requisitos la cual se definirá a continuación.

Ingeniería de Requisitos

Montilva y otros (2008), definen la Ingeniería de Requisitos, como un enfoque sistemático para descubrir, organizar y documentar los requisitos de un sistema. Es una disciplina dentro de la Ingeniería de Software encargada del estudio de los requisitos en los sistemas, estableciendo principios, modelos, métodos, técnicas y herramientas automatizadas que contribuyan a mejorar la definición y especificación de requisitos, lo cual permite lograr un entendimiento común entre clientes/usuarios e ingenieros de software sobre los requisitos que debe satisfacer la aplicación.

Al respecto, Pressman (2010) afirma:

> La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. (p. 120)

De este modo es importante destacar, que en todo el proceso de la Ingeniería de requisitos, el ingeniero de software se comunica con un conjunto de participantes diferentes pero los clientes y los usuarios tiene el impacto más significativo sobre el trabajo técnico que se realiza. En algunos casos el cliente y usuario son una misma persona, pero en muchos proyectos el cliente y el usuario son personas diferentes, en este sentido Pressman los define de la siguiente manera:

- **Cliente**: es aquel que solicita el software que se va a construir, define los objetivos generales de negocios para el software, proporciona los requisitos básicos del producto y coordina los recursos económicos para el proyecto.
- **Usuario o Usuario Final:** es la persona o grupo de personas que en realidad usarán el software que se va a construir para alcanzar algún propósito de negocios, definirá los detalles operativos del software de forma que el propósito del negocio pueda alcanzarse. Así, para el caso de la presente investigación el método a desarrollar orientado a validar los requisitos funcionales de software a partir de prototipos, se realizará en sesiones de validación con el usuario final.

En este orden de ideas, Montilva y otros, sostienen que el proceso de ingeniería de requisitos puede ser descrito en cinco procesos, tal como se describió en la figura 1, los tres primeros procesos son secuenciales y los dos últimos se realizan a lo largo de los tres primeros. Cada uno de estos procesos se describe a continuación:

1. **Descubrimiento e Identificación de Requisitos:** Pressman (2010), afirma que se refiere al proceso de elicitación o descubrimiento de los requisitos directamente con el cliente, los usuarios y los que están involucrados en los objetivos del sistema o producto y sean expertos. Investigar cómo los sistemas o productos se ajustan a las necesidades del negocio, y finalmente, cómo el sistema o producto va a ser utilizado en el día a día. Este proceso que resulta simple puede ser muy complicado.

Christel y Kang (1992) identifican una serie de problemas que ayudan a comprender por qué la obtención de requisitos es costosa, entre ellos se encuentran:

- Problemas de alcance: el límite del sistema está mal definido o los detalles técnicos innecesarios, que han sido aportados por los clientes/usuarios, pueden confundir más que clarificar los objetivos del sistema.
- Problemas de comprensión: los clientes/usuarios no están completamente seguros de lo que necesitan, tienen una pobre compresión de las capacidades y limitaciones de su entorno de computación, no existe un total entendimiento del

dominio del problema, existen dificultades para comunicar las necesidades al ingeniero del sistema, frecuentemente omiten información por considerar que es "obvia", especifican requisitos que están en conflicto con las necesidades de otros usuarios, o especifican requisitos ambiguos o poco estables

- Problemas de volatilidad: los requisitos cambian con el tiempo y hay que adaptarse a esos cambios.

Según Pressman, para ayudar a solucionar estos problemas, los ingenieros de sistemas deben aproximarse de una manera organizada a clientes/usuarios para definir e identificar requisitos, donde el resultado alcanzado como consecuencia de la identificación de requisitos variará dependiendo del tamaño del sistema o producto a construir. Para grandes sistemas, el producto obtenido debe incluir:

- Una relación de necesidades y características;
- Un informe conciso del alcance del sistema o producto;
- Una lista de clientes, usuarios y otros intervinientes que deben participar en la actividad de obtención de requisitos;
- Una descripción del entorno técnico del sistema; una relación de requisitos (perfectamente agrupados por funcionalidad) y las restricciones del dominio aplicables a cada uno;
- Un conjunto de escenarios que permiten profundizar en el uso del sistema o producto bajo diferentes condiciones operativas, y
- Un prototipo desarrollado para definir mejor los requisitos.

Cada uno de los productos obtenidos debe ser revisado por las personas que hayan participado en la obtención de sus requisitos. En este orden de ideas, según Pressman (2010) las técnicas empleadas en el proceso de descubrimiento de requisitos son las entrevistas, casos de uso, reuniones, observación, reuso de requisitos, modelado de sistemas, entre otros. Para el interés del presente trabajo de investigación se detallara los escenarios de usuario, o casos de uso a continuación.

Escenarios del usuario: una vez recopilados los requisitos, bien sea por observación, reuniones o entrevistas, el ingeniero del software (analista) puede crear un conjunto de escenarios que identifiquen una línea de utilización para el sistema que va a ser construido. Los escenarios, algunas veces llamados casos de uso, proporcionan una descripción de cómo el sistema se usará (Pressman, 2010).

De este modo, para Berenbach, Paulish, Kazmeier y Rudorfer (2009), los casos de uso son técnicas que facilitan la captura y documentación de las funciones del sistema desde el punto de vista de sus usuarios directos. Describen gráfica y textualmente la funcionalidad del sistema, el conjunto de funciones que el sistema ofrecerá al usuario a través de su interfaz.

Para crear un caso de uso, el analista debe primero identificar los diferentes tipos de actores que utilizarán el sistema o producto. Estos actores representan roles que van a interactuar con el sistema. Definido más formalmente, un actor es aquel (persona, dispositivo, otros sistema) que se comunica con el sistema o producto y que es externo al sistema en sí mismo, es aquel que lleva a cabo un rol en el sistema. Así un usuario final puede tener varios roles (Pressman, 2010).

Una vez que se han identificado los actores, se pueden desarrollar los casos de uso. Los casos de uso especifican la manera en que los actores interactúan con el sistema, y describen los escenarios que serán percibidos de distinta forma por distintos actores. Jacobson (1993), sugiere un número de preguntas que deberán responderse por el caso de uso:

- ¿Cuáles son las principales tareas o funciones que serán realizadas por el actor?
- ¿Cuál es el sistema de información que el actor adquiere, produce o cambia?
- ¿Qué actor informará al sistema de los cambios en el entorno externo?
- ¿Qué información necesita el actor sobre el sistema?

2. **Análisis de Requisitos y Negociación:** es el proceso mediante el cual se razonan y analizan las necesidades identificadas de los clientes y usuarios para llegar a una definición de requisitos de la aplicación, Montilva y otros (2008), ver figura 8. Una

vez recopilados los requisitos, el producto obtenido configura la base del análisis de requisitos. Los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud y ambigüedad, y se clasifican en base a las necesidades de los clientes/usuarios.

Es común en clientes y usuarios solicitar más de lo que puede realizarse, consumiendo recursos de negocio limitados, así como también en ocasiones proponen requisitos contradictorios; para estos casos el Ingeniero del Sistemas debe resolver estos conflictos a través de un proceso de negociación. Los clientes, usuarios y el resto de intervinientes deberán clasificar sus requisitos y discutir los posibles conflictos según su prioridad. Utilizando un proceso de análisis de requisitos iterativo, se irán eliminando, combinando y/o modificando requisitos para conseguir satisfacer los objetivos planteados.

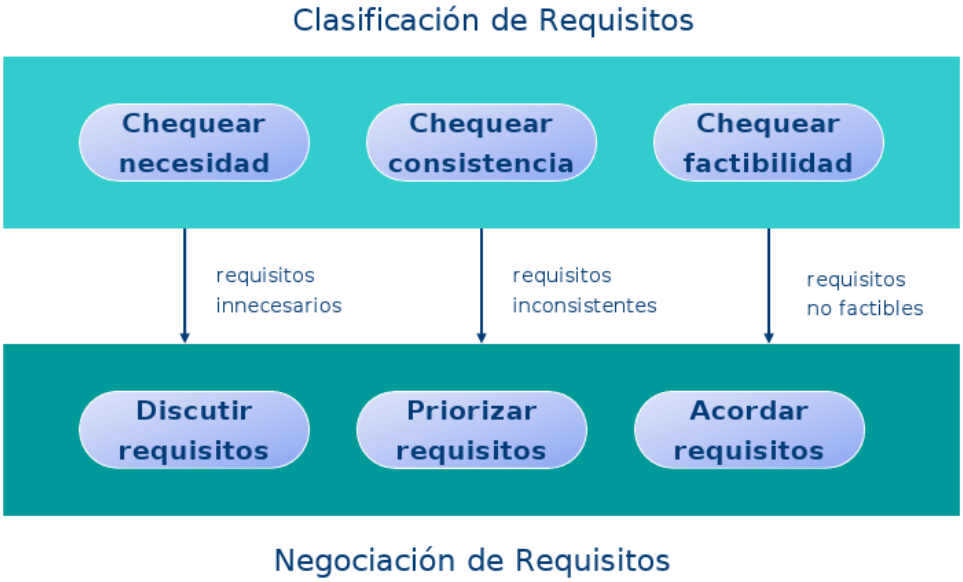

Figura 8. Actividades de Negociación de Requisitos y su Relación con la Clasificación. Fuente: Montilva y otros (2008)

3. **Especificación de Requisitos:** Pressman (2010) lo define como el producto final sobre los requisitos del sistema obtenidos por el ingeniero, es el proceso mediante el

cual los requisitos definidos por clientes y usuarios se documentan. El término especificación puede significar distintas cosas para diferentes personas. Una especificación puede ser un documento escrito, un modelo gráfico, un modelo matemático formal, una colección de escenarios de uso, un prototipo o una combinación de lo anteriormente citado.

Algunos autores sugieren que debe utilizarse una "plantilla estándar" y usarse en la especificación de requisitos, argumentando que así se conseguirían requisitos que sean presentados de una forma más consistente y más comprensible, un documento con una descripción detallada de los requisitos que la aplicación debe satisfacer. De esta manera el documento de requisitos (DR) es uno de los productos principales en el proceso de Ingeniería de Requisitos, el cual según Montilva y otros, es utilizado para:

- Registrar y validar los requisitos de los clientes y usuarios del sistema
- Especificar las características técnicas del sistema a fin de facilitar su diseño
- Facilitar las actividades de verificación y validación del sistema, adiestramiento de los usuarios del sistema y mantenimiento del sistema.

No obstante, en muchas ocasiones es necesario buscar la flexibilidad cuando una especificación va a ser desarrollada. Para grandes sistemas, un documento de requisitos, combinado con descripciones en lenguaje natural, casos de uso y modelos gráficos puede ser la mejor alternativa. En cualquier caso, los escenarios a utilizar pueden ser tanto los requeridos para productos de tamaño pequeño o los de sistemas que residan en entornos técnicos bien conocidos.

4. **Validación de Requisitos:** el resultado del trabajo realizado es una consecuencia de la ingeniería de requisitos (especificación del sistema e información relacionada) y es evaluada su calidad en la fase de validación. Esta fase será detallará a continuación.

5. **Gestión de Requisitos:** los requisitos del sistema cambian a lo largo de la vida del sistema. La gestión de requisitos es un conjunto de actividades que ayudan al equipo

de trabajo a identificar, controlar y seguir los requisitos y los cambios que estos sufran en cualquier momento. Este proceso maneja los cambios en los requisitos una vez que estos hayan sido verificados y validados y se asegura que todos los requisitos sean implementados. En este sentido, Sommerville (2005) define Gestión de Requisitos como el proceso de entender y controlar los cambios en los requisitos, el objetivo de la gestión de requisitos es que los cambios se realicen de forma consistente haciendo posible el control de versiones.

Validación de Requisitos

En el proceso de validación de requisitos se examinan las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto (Pressman, 2010).

El equipo de revisión incluye ingenieros del sistema, clientes, usuarios, expertos del dominio y otros intervinientes que examinan la especificación del sistema buscando errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta, inconsistencias, requisitos contradictorios, o requisitos imposibles o inalcanzables. Para ello en este proceso de verificación y validación se emplean una o más de estas actividades: revisión de requisitos, elaboración de prototipos, validación de modelos, diseño de pruebas de aceptación, entre otros.

De este modo, según Sommerville (2005) no deben subestimarse los problemas en la validación de requisitos, es difícil demostrar que un conjunto de requisitos cumple las necesidades del usuario. Los usuarios deben imaginarse al sistema en funcionamiento y cómo este encajaría en su trabajo. Para los ingenieros de software puede ser difícil llevar a cabo este tipo de análisis abstracto, pero para los usuarios del sistema es aún más difícil. Es por ello que en algunas ocasiones, un buen aliado del ingeniero de requisitos para validar requisitos es el uso de prototipos de interfaz de usuario, el cual se detalla a continuación.

Prototipos de Interfaz

Sommerville (2005), sostiene que debido a la naturaleza dinámica de las interfaces de usuario, las descripciones textuales y los diagramas que se obtienen como resultado de la fase de Especificación de Requisitos del proceso de Ingeniería de Requisitos previamente mencionado, no son suficientes ni adecuadas para expresar los requisitos de estas interfaces. Es por ello que la construcción de prototipos con la implicación de los usuarios finales es la única forma práctica de diseñar y desarrollar interfaces gráficas de usuario para sistemas de software. Implicar al usuario en el proceso de diseño y desarrollo es un aspecto fundamental en el criterio de diseño para sistemas interactivos.

Por lo tanto, Sommerville define prototipo como una versión inicial de un sistema de software que se utiliza para demostrar conceptos, probar opciones de diseño y, en general, informarse más del problema y sus posibles soluciones.

En este orden de ideas, Kendall y Kendall (1995) afirman que "Los prototipos son una visión preliminar del sistema futuro". (p. 197). La elaboración de prototipos de un sistema de información es una técnica valiosa para la recopilación rápida de información específica acerca de los requisitos de los usuarios.

De esta forma, según Kendall y Kendall, a través de la elaboración del prototipo el ingeniero de software está buscando

- Las reacciones iniciales de los usuarios ante el prototipo: en este punto es importante detallar la manera en que reacciona el usuario al trabajar con el prototipo, y qué tan buen ajuste hay entre sus necesidades y las características del prototipo del sistema. Las reacciones son recopiladas por medio de observaciones, entrevistas y formas de retroalimentación (posiblemente cuestionarios) diseñadas para recoger la opinión de cada persona acerca del prototipo cuando interactúa con él. Por medio de tales reacciones de usuario, el analista descubre muchas perspectivas en el prototipo. También es importante recopilar las sugerencias de los usuarios acerca de cómo refinar o cambiar el prototipo presentado.

- Innovaciones: las innovaciones para el prototipo son un punto importante en la captura de requisitos, ya que, de ser satisfactorias, serán parte del sistema. Las innovaciones son capacidades nuevas del sistema que no habían sido pensadas o descubiertas por el usuario antes de la interacción con el prototipo, estas generalmente añaden algo nuevo e innovador.
- Corrección de errores u omisiones: el usuario puede detectar si los requisitos indicados fueron entendidos y plasmados, y pueden detectar fácilmente si hay algún error u omisión de los requisitos planteados.

Por lo anteriormente expuesto, los usuarios tienen un papel importante en el proceso de elaboración de prototipos. Su primer interés debe ser interactuar con el prototipo mediante experimentación. Los analistas de sistemas deben trabajar sistemáticamente para obtener y evaluar las reacciones de los usuarios ante el prototipo, y luego trabajar para incorporar las sugerencias e innovaciones de los usuarios que valgan la pena en las modificaciones subsecuentes.

Por lo tanto el uso de prototipos de interfaz de usuarios es de vital importancia en la Ingeniería de Requisitos, ya que ayuda a garantizar el éxito de la misma, permitiendo validar eficazmente los requisitos obtenidos y obtener nuevas necesidades que surjan en la revisión del prototipo.

Aplicaciones Web

Internet y la Web han influido enormemente en el mundo de la informática como en la sociedad en general., en los últimos años ha transformado los sistemas informáticos: ha roto las barreras físicas, económicas y lógicas y ha abierto un abanico de nuevas posibilidades. Una de las áreas que mas expansión y popularidad ha tenido en la Web en los últimos años son las aplicaciones Web, debido a lo práctico que supone que el navegador Web sea un cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar

software a miles de usuarios potenciales (Rossi y otros, 2010).

La W3C[10] define el término Web como "el universo de información accesible a través de la red". Una aplicación Web es un sistema que permite a un usuario final acceder a una parcela de información contenida en el universo al que hace referencia la anterior definición del W3C.

Así, Lujan (2002) define una aplicación Web como una aplicación dinámica basada en la arquitectura cliente/servidor, donde el cliente es un navegador Web y el servidor es un servidor Web, utilizando ambos para su entendimiento el protocolo de aplicación HTTP[11]. El protocolo HTTP forma parte de la familia de protocolos de comunicaciones TCP/IP[12], que son los empleados en Internet. Estos protocolos permiten la conexión de sistemas heterogéneos, lo que facilita el intercambio de información entre distintos ordenadores.

Covella (2005) afirma que los sitios y aplicaciones Web involucran una mezcla entre publicación de contenidos impresos y desarrollo de software, entre marketing y computación, entre comunicaciones internas relaciones externas, y entre arte y tecnología. Por ello, el desarrollo de Aplicaciones Web posee determinadas características que lo diferencian del desarrollo de aplicaciones o software tradicional tales como:

- Son evolutivas, tanto en sus requerimientos como en su funcionalidad. Están pensadas para diferentes públicos, los cuales tienen distintas necesidades y habilidades.
- Deben presentar diversos tipos de contenido (texto, imágenes, video, audio, presentaciones, entre otros).
- Estéticamente atractivas y disponer de un diseño de navegación sencillo e intuitivo.
- Deben considerar estándares y usos culturales y sociales que permitan su internacionalización.

---

[10] W3C: Word Wide Web Consortium, http://www.w3.org/

[11] HTTP: HyperText Transfer Protocol, http://www.w3.org/Protocols/

[12] TCP/IP: Protocolo de control de transmisión/Protocolo de Internet, http://www.w3.org/Protocols/HTTP/AsImplemented.html

- Contemplan características de seguridad y privacidad de datos.
- Deben estar desarrolladas teniendo presentes los diversos tipos de formatos necesarios según las plataformas (celulares, PDAs, entre otros).

En este orden de ideas, según Campana (2009) las aplicaciones Web tradicionales son aquellas que presentan documentos HTML planos en un navegador y reciben de vuelta peticiones HTTP. En las aplicaciones Web de tipo cliente delgado (thin-client) no existe código del lado del cliente, solamente HTML. La figura 9 muestra la interacción entre un cliente y un servidor, sincrónica, mediante el envío de peticiones HTTP y su posterior respuesta, repitiéndose la secuencia en el tiempo.

De este modo a medida que fueron creciendo las necesidades, las aplicaciones Web se orientaron hacia el usuario, surgiendo lo que se denominó la Web 2.0, la cual es la siguiente generación Web en donde las aplicaciones son más interactivas. El término Web 2.0, fue acuñado por O'Reilly, (O'Reilly y Battelle, 2009), está asociado a aplicaciones Web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la World Wide Web. Se refiere a una nueva generación de aplicaciones Web que provee participación, colaboración e interacción en línea a los usuarios.

En contraste con la Web tradicional, Web 2.0 ofrece más que interacción básica y participación de usuarios, se utilizan interfaces dinámicas y atractivas que se acercan a las aplicaciones de escritorio. De allí surgen las Rich Internet Applications (RIA), que forman un nuevo tipo de experiencia del usuario de Internet que es más interactiva, de mejor respuesta y más atractivas que las iteraciones con aplicaciones Web HTML tradicionales. Las RIA combinan las ventajas de la interactividad directa y respuesta inmediata común a las tradicionales aplicaciones de escritorio con la amplia distribución de las aplicaciones Web. (Rooms, 2007).

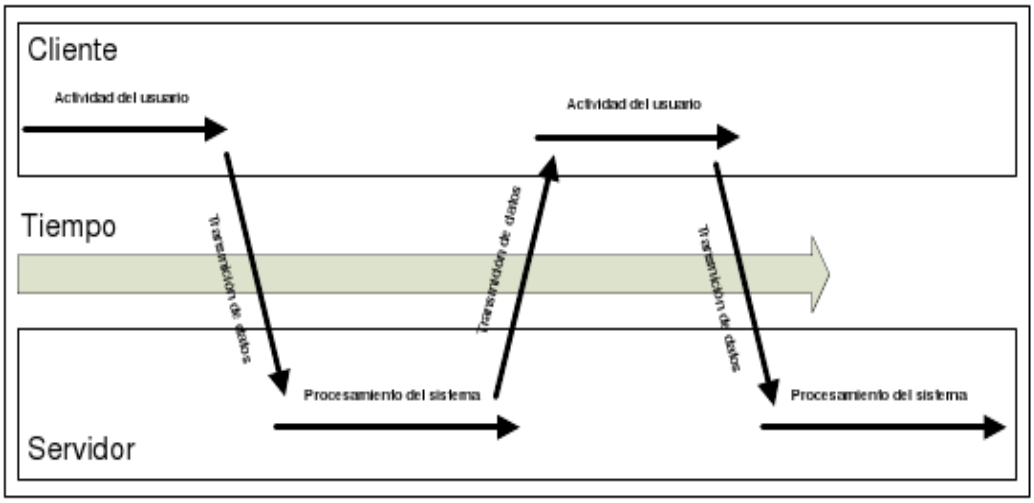

Figura 9. Procesamiento de Datos en Aplicaciones Web Tradicionales
Fuente: Garret (2005)

RIA. *Rich Internet Application*

Rossi y otros (2010) definen *Rich Internet Application*, como un nuevo tipo de aplicación Web cuyo objetivo es el de incrementar y mejorar las opciones y capacidades de las aplicaciones Web tradicionales. Este tipo de aplicaciones son desarrolladas, en la mayoría de los casos, utilizando lenguajes basados en XML, y son ejecutadas utilizando unos servidores de presentación también propios. Las limitaciones en la capa de presentación de los actuales navegadores Web y del lenguaje HTML ha sido lo que ha impulsado a los desarrolladores a utilizar este nuevo tipo de aplicaciones, que permiten, entre otras cosas, mejorar la experiencia entre el usuario y la aplicación, la ejecución de contenido multimedia y la carga de aplicaciones online/offline, dependiendo de la tecnología RIA que se utilice.

Según Farré (2005) las RIA cumplen, la mayoría, con una serie de características elementales que son las siguientes:

- Una alta interactividad con el usuario, a través multitud de elementos de interacción que antes sólo eran viables en entornos de escritorio (como menús de navegación, árboles, deslizadores, etc.), programables bajo cualquier evento de usuario (como

clic de ratón, pulsación de tecla, arrastrar y soltar, entre otros).

- Una alta velocidad de respuesta a la interacción del usuario. La unidad de información mínima es la que desee el desarrollador: una etiqueta, una tabla o quizás toda la página. De este modo, es posible conseguir una respuesta inmediata a las acciones del usuario, descargando sólo los datos absolutamente necesarios.
- La complejidad de desarrollo de las aplicaciones no difieren mucho de las aplicaciones Web existentes.
- La aplicación realiza todo lo anteriormente dicho sin la necesidad de grandes plataformas y compatible con todos los navegadores.

La figura 10 muestra el procesamiento de datos en una aplicación RIA, si bien existe comunicación entre el cliente y el servidor para realizar el procesamiento, esta interacción es menor que en las aplicaciones Web tradicionales (ver figura 9). Las RIA están basadas en AJAX[13], donde las aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, de esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Farré (2005) afirma que existen varias tecnologías que permiten la creación de RIA, capaces de introducir en la Web el paradigma de la aplicación de escritorio. Todas ellas cumplen las características anteriores y además, su funcionamiento se basa en los mismos principios:

- Todas ellas utilizan un modelo contenedor en el lado del cliente que almacena la parte gráfica (esquema) de la aplicación. Con esto se reduce considerablemente la comunicación con el servidor y evita tener que renderizar una nueva página Web a cada clic del usuario.

[13] AJAX: Asynchronous Javascript And XML, http://www.w3schools.com/ajax/

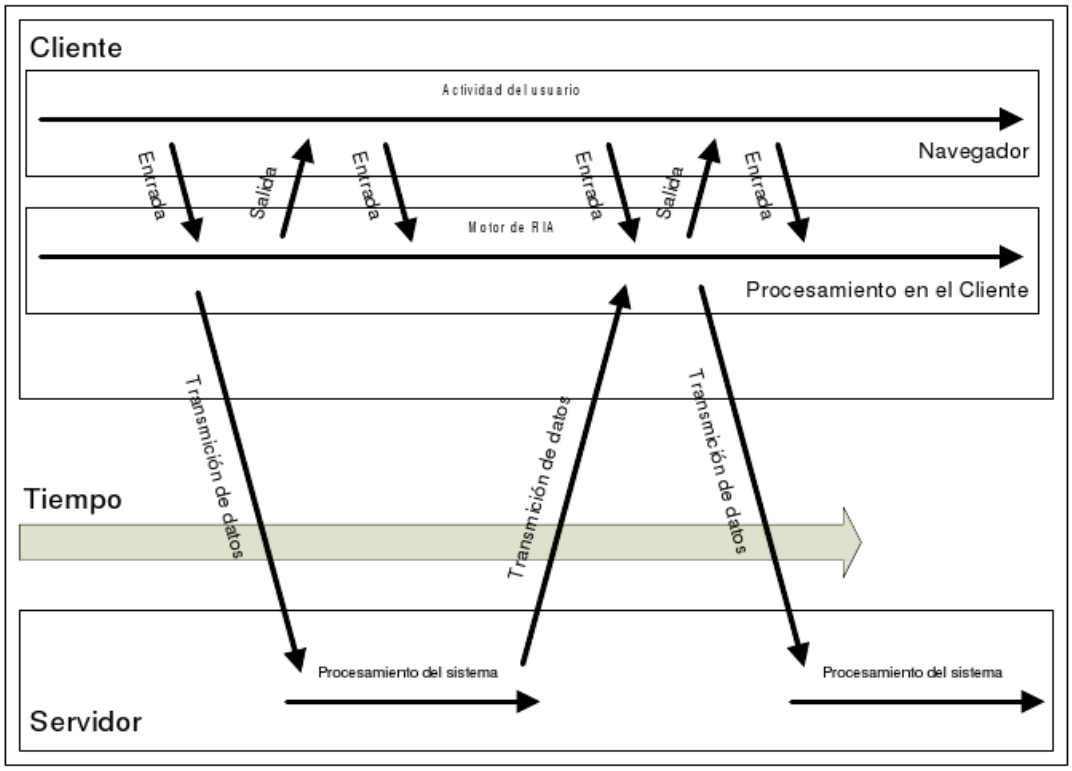

Figura 10. Procesamiento de Datos en Aplicaciones RIA.
Fuente: Garrett (2005)

- La mayoría de ellas permiten el desarrollo de las aplicaciones a través del navegador Web.
- Utilizan un lenguaje basado en XML para definir las interfaces de usuario.

Patrones RIA

Scott y Neil (2009), definen los patrones RIA como patrones de diseño Web, donde un patrón de diseño Web es una descripción de un problema de diseño Web y buenas soluciones a este problema, donde el problema que viene a solucionar el patrón RIA es un problema de interfaz de usuario, de interacción, usabilidad o prestación en la misma, para lograr mejorar tanto la funcionalidad del sitio como la amigabilidad del mismo.

En este sentido, según Koch y otros (2009), los patrones RIA describen la

interacción, operación y presentación de widgets[14] de RIA. Cada patrón comienza con un evento de usuario o un evento del sistema, es decir una interacción, por ejemplo, colocar el punto del ratón sobre el componente, colocar el foco en un componente, presionar alguna tecla, click del ratón, entre otros. Así una variedad de operaciones pueden ser desencadenadas por la interacción, tales como validar, buscar, autocompletar o refrescar información. Finalmente, el resultado de la operación implica una actualización de la interfaz de usuario.

De este modo, Scott y Neil, proponen seis (6) principios de diseño simples para agrupar los patrones y las mejores prácticas para la creación de aplicaciones de internet enriquecidas, estos principios son:

1. **Ser directo (Make it Direct):** el principio de la manipulación directa se basa en el hecho de que en lugar de ir a una página independiente para editar el contenido se puede hacer en línea, directamente en el contexto de la página, (ver figura 11).

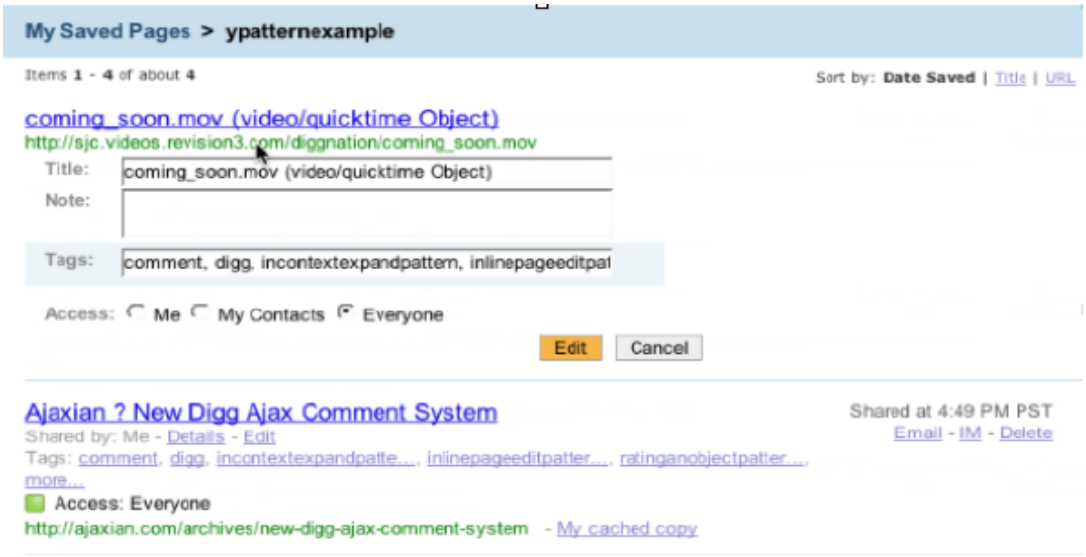

Figura 11. Ejemplo Principio Make it Direct
Fuente: Scott y Neil (2009)

[14] Widget: es un componente, o un control de una interfaz gráfica de usuario, con el cual el usuario interactúa para llevar a cabo una serie de comandos, http://www.widgets.yahoo.com/

Los patrones de interacción que se pueden utilizar para hacer la interfaz más directa son:

- *In page editing*: editar contenido directamente en la página.
- *Drag and drop*: mover objetos con la ayuda del ratón.
- *Direct selection*: aplicar acciones directamente sobre el objeto seleccionado.

2. **Mantenerlo ligero (*Keep It Lightweight):*** Cuando el usuario interactúa con la aplicación debe ser de una manera que cause el mínimo de esfuerzo físico y mental, (ver figura 12). Dentro de este principio se encuentra el patrón:
   - Contextual Tools: herramientas contextuales que permiten en lugar de interactuar con alguna funcionalidad por separado, esta se debe incorporar en el contenido directamente. Se podría decir que son la versión en Web, de menús del botón derecho del escritorio, pero en lugar de presionar el botón derecho del ratón para que aparezca un menú, este se puede revelar mediante herramientas en el contexto del contenido, a través de: herramientas siempre visibles, multiniveles o menús secundarios.

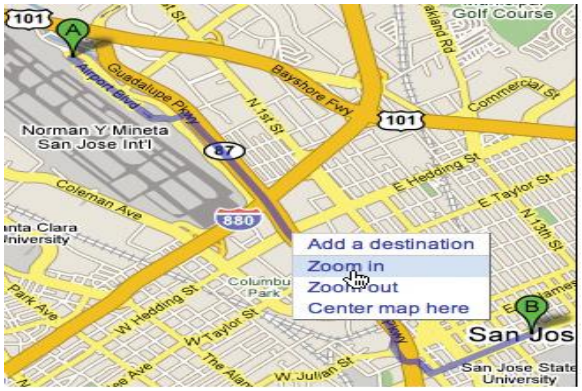

Figura 12. Ejemplo Principio Keep It Lightweight
Fuente: Scott y Neil (2009)

3. **Permanecer en la página (*Stay on the Page*):** La actualización de la página, es decir recargar la página completa durante la interacción con la aplicación Web, suele ser perjudicial para el flujo mental del usuario. Por defecto el flujo de

navegación de las aplicaciones Web es ir de página a página por cada acción. Ahora que ya no existente limitaciones técnicas, se puede decidir cuándo mantener al usuario en la página y el flujo de la navegación, (ver figura 13). Los patrones que permiten mantener al usuario en la página son los siguientes:

- *Overlays*: en lugar de ir a una nueva página, un mini-página se pueden mostrar en una capa ligera sobre la página.
- *Inlays*: en lugar de ir a una nueva página, la información o acciones pueden ser incrustados dentro de la página.
- *Virtual Pages*: al revelar el contenido dinámico y el uso de la animación, se puede ampliar el espacio virtual de la página.
- *Process Flows:* en lugar de pasar de una página a otra, se puede crear un flujo dentro de una misma página.

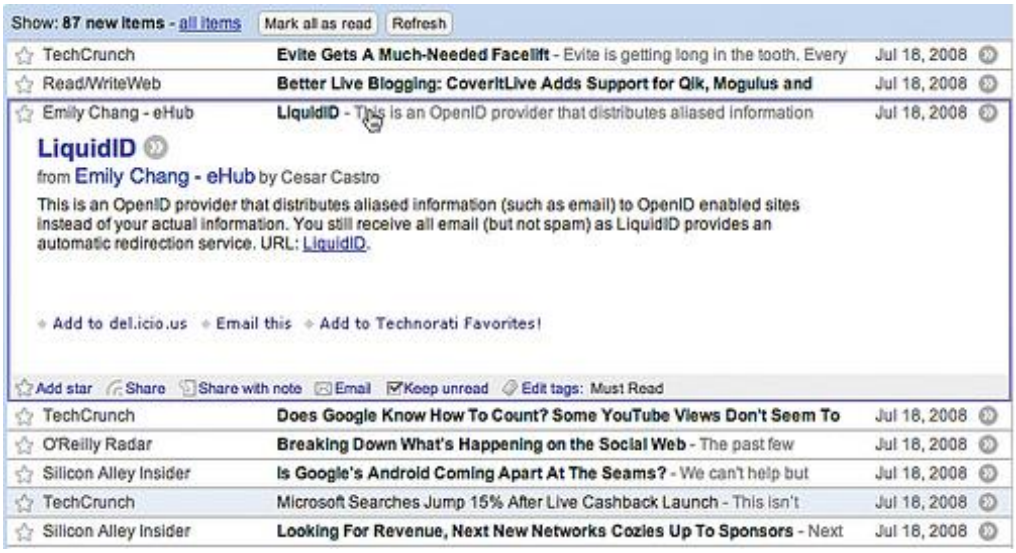

Figura 13. Ejemplo Principio Stay on the Page
Fuente: Scott y Neil (2009)

4. **Proporcionar una invitación (*Provide an Invitation*):** para los usuarios que no están acostumbrados a interactuar con las características que proporcionan las RIA, muchas veces desconocen que existen funcionalidades en la interfaces que pueden mejorar su experiencia, es por ello que el descubrimiento es uno de los

principales desafíos para una interacción enriquecida en la Web. Una característica RIA no es de utilidad si los usuarios no lo descubren, es por ello que una manera clave de mejorar el descubrimiento es proporcionar las invitaciones, es decir dar pistas para la interacción. Las invitaciones indican al usuario que existe un nuevo nivel de interacción, (ver figura 14), para ello existen dos (2) patrones:

- *Static invitations*: estas son las invitaciones que se ofrecen en la página utilizando técnicas visuales para invitar a la interacción.
- *Dinamic invitations*: estas invitaciones entran en juego en respuesta a qué y donde este interactuando el usuario.

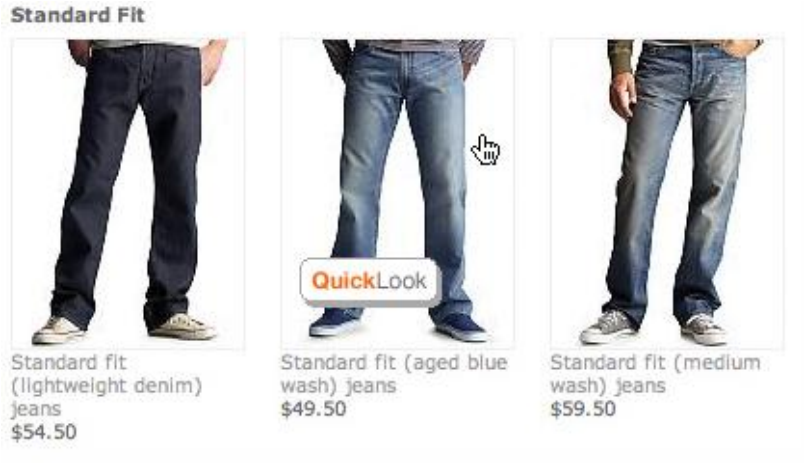

Figura 14. Ejemplo Principio Provide an Invitation
Fuente: Scott y Neil (2009)

5. **Usar transiciones (*Use Transitions*):** Las animaciones, efectos cinemáticos, y otros tipos de transiciones visuales pueden ser técnicas de gran alcance para mejorar la participación y la comunicación con los usuarios, (ver figura 15). Así los patrones que permiten esta transición son:
   - *Brighten and Dim:* Iluminar un espacio de la pantalla para lograr la atención del usuario allí.
   - *Expand/Collapse*: es útil disponer de contenido adicional u otros paneles ocultos hasta que el usuario los necesite. Esto se logra mediante el uso de expandir y contraer para controlar la visibilidad de un contenido en el flujo de la

página.

- *Self-Healing Fade*: al eliminar o mover elementos, a menudo es útil exponer temporalmente un "espacio vacío" donde se eliminó el objeto, así este efecto se logra con la animación de al eliminar un ítem, cerrar el espacio donde ese elemento ha sido eliminado.
- *Animation:* efectos de movimiento de objetos dentro de la página.
- *Spotlights*: son útiles cuando se ha producido un cambio en la interfaz, por un momento se destaca un objeto, que sutilmente se puede notificar al usuario de un cambio en la interfaz.

Figura 15. Ejemplo Principio Use Transitions
Fuente: Scott y Neil (2009)

6. **Reaccionar Inmediatamente (*React Immediately*):** Una interfaz que proporcione una respuesta es una interfaz inteligente. Este principio explora cómo realizar una experiencia enriquecida mediante el uso de las respuestas animadas, (ver figura 16). Una buena parte de las interfaces de aplicación está involucrada en la búsqueda de información, si una aplicación está realizando búsquedas directas, filtrando los resultados de búsqueda, o ayudan en la entrada, hay muchas maneras de proporcionar asistencia de búsqueda. Para ello hay cuatro patrones de búsqueda pertenecientes al principio de reaccionar inmediatamente, estos son:

- *Auto Complete*: proporciona ayuda al usuario para completar su escritura en un campo de entrada, a través de un menú desplegable de valores coincidentes.
- *Live Suggest*, *Live Search*, *Refining Search*: ofrecen sugerencias en términos de búsqueda en tiempo real para la creación nuevas búsquedas y obtener resultados en tiempo real.

De este modo, el principio reaccionar inmediatamente, no solo se refiere a los patrones de búsqueda, también permite proporcionar retroalimentación interactiva en otras situaciones. Hay varios patrones que ayudan a mantener al usuario informado sobre lo que está sucediendo en la aplicación, ellos son:

- *Live Preview*: previsualización dinámica ofrece a los usuarios una idea de antemano de cómo la aplicación va a interpretar su entrada una vez presentada.
- Progressive Disclosure, Progress Indicator: cuando la aplicación está ocupada con un proceso muy largo, es necesario mantener informado al usuario del progreso del proceso.
- *Periodic Refresh*: en ocasiones una interfaz debe mostrar el contenido nuevo o las actualizaciones que se pueden haber producido en la aplicación por otros usuarios. En tal caso, la interfaz no está realmente reaccionando a la entrada de un usuario, sino a la comunidad más grande en su conjunto. La actualización periódica aporta un nuevo contenido de forma periódica sin la interacción directa del usuario.

Figura 16 .Ejemplo Principio React Immediately
Fuente: Scott y Neil (2009)

De esta manera con el uso de estos patrones, se garantiza una experiencia enriquecida para el usuario, proporcionándole usabilidad, facilidad de interacción y de aprendizaje, amigabilidad, entre otros. Estas características ayudarán a lograr alcanzar la calidad de una aplicación Web. En el presente trabajo de investigación cuyo objetivo es desarrollar un método de validación de requisitos funcionales a través de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad, se hace necesario definir los términos relacionados con el modelo de calidad a utilizar, los cuales se exponen a continuación.

Modelo de Calidad de Software

La importancia de la calidad de software en la disciplina de la Ingeniería del software es ampliamente reconocida en la actualidad, de allí su gran importancia, se entiende por calidad de software: "al grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario". (IEEE, 1990, p.60).

Según Pressman (2010), la calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. Para Callaos y Callaos (1993), la calidad del software no es algo que depende de una sola característica en particular, sino que obedece al compromiso de todas sus partes, valorándose la calidad del producto y la calidad del proceso. Es relevante valorar, que la funcionalidad de un producto ha sido la única manera de medir la calidad del software. Es por ello que se hace necesaria la adopción de un estándar de calidad, modelos estándar que establecieran las características fundamentales de calidad del software y que ganara el consenso para su aplicación.

Así, según ISO/IEC 9126 (1991), "un modelo de calidad es la descripción, en términos de un conjunto estructurado de características y sub-características, de los

atributos que debe tener un producto software y que contribuyen a que dicho producto sea bueno dentro de los de su clase". Los modelos de calidad presentan un acercamiento para unir diferentes atributos de calidad con los objetivos básicos de: ayudar a entender como varias facetas de calidad contribuyen al todo. Además los modelos representan una ayuda para navegar por el mapa de características de calidad, sub-características y medidas apropiadas, ayudando a definir el perfil de evaluación. Moreno, Bolaños y Navia (2010).

Algunos de los modelos y estándares de calidad más usados y que han sido referenciados en una apreciable cantidad de trabajos son, según Moreno, Bolaños y Navia (2010): McCall, Boehm, FURPS, ISO 9126. En el presente trabajo de investigación se tomará en cuenta el modelo de calidad McCall y el modelo ISO 9126-1, según los factores de calidad resultantes de la combinación de ambos modelos que se realizo en el método propuesto por Macías y Gómez (2010), el cual se especificó anteriormente. Estos factores de calidad serán incorporarlos en el método de validación de requisitos a partir de prototipos de interfaces de usuario basado en patrones RIA a presentar, tomando como base estos factores se evaluará la calidad del prototipo por parte de los usuarios, lo cual ayudará en el proceso de validación de requisitos. A continuación se describen el modelo de calidad McCall e ISO 9126-1.

Modelo McCall

Este modelo fue propuesto por McCall en 1977, está orientado a los desarrolladores de sistemas, para ser utilizado durante el proceso de desarrollo para medir la calidad del software (Pressman, 2010). El modelo McCall organiza los factores de calidad en tres áreas de trabajo desde los cuales el usuario puede contemplar la calidad de un producto, basándose en once factores de calidad organizados en torno a esas áreas de trabajo, tal como se muestra en la siguiente figura:

Figura 17. Factores de Calidad de McCall
Fuente: Adaptado de Presmman (2010)

A continuación se describen cada uno de los factores de calidad definidos por McCall, según su clasificación (Pressman, 2010)

- Operación del Producto: requiere que pueda ser comprendida rápidamente, operada eficientemente y que los resultados sean aquellos requeridos por el usuario.
  - Corrección: el grado en el que el programa cumple con su especificación y logra los objetivos que propuso el cliente.
  - Confiabilidad: el grado en el que se espera que un programa lleve a cabo su función con la exactitud precisión.
  - Eficiencia: la cantidad de recursos informáticos y código necesario para que un programa realice su función.
  - Integridad: el grado de control del acceso del software a personas no autorizadas.
  - Facilidad de Uso: el esfuerzo necesario para aprender, operar, preparar los datos de entrada e interpretar las salidas (resultados) de un programa.
- Revisión del Producto: Está relacionada con la corrección de errores y la adaptación de los sistemas. Esto es importante porque es generalmente considerada como la parte más costosa en el desarrollo de software.
  - Facilidad de Mantenimiento: el esfuerzo necesario para localizar y corregir un

error del programa.

- Flexibilidad: el esfuerzo necesario para modificar un programa en operación
- Facilidad de Prueba: el esfuerzo que demanda probar un programa con el fin de asegurar que realice su función

- Transición del Producto: Puede que no sea muy importante en todas las aplicaciones. Sin embargo, la orientación a procesamiento distribuido y el rápido cambio en el hardware es probable que incremente su importancia
  - Portabilidad: el esfuerzo necesario para transferir el programa de un entorno de sistema de hardware y/o software a otro.
  - Facilidad de Reutilización: el grado en el que un programa o parte de él pueda reutilizarse en otras aplicaciones.
  - Interoperativilidad: el esfuerzo necesario para acoplar un sistema con otro.

Modelo ISO 9126 -1 (1991/2001)

El estándar internacional ISO 9126 se publicó en 1991, tiene como objetivo la definición de un modelo de calidad y su uso como marco para la evaluación de software. ISO/IEC 9126 (1991). El estándar ISO-9126 define un modelo, basado en modelos ya existentes como McCall y Boehm, el cual presenta dos partes:

- Modelo de calidad para calidad externa e interna.
- Modelo de calidad para calidad en uso.

La calidad interna tiene como objetivo medir la calidad del software mediante factores medibles durante su desarrollo. La calidad externa pretende medir la calidad del software teniendo en cuenta el comportamiento de este software en un sistema del cual forme parte. Finalmente, la calidad en uso corresponde a la calidad del software desde el punto de vista de un usuario. La versión de 2001 de la norma ISO 9126 está dividida en cuatro partes, ISO/IEC 9126-1 (2001).

- ISO 9126-1. Modelo de calidad.

- ISO 9126-2. Métricas externas.
- ISO 9126-3. Métricas internas.
- ISO 9126-4. Calidad en las métricas de uso.

De este modo, la calidad de un producto software debería ser evaluado usando un modelo de calidad. ISO 9126-1 propone un modelo de calidad categorizando la calidad de los atributos software en características o factores, las cuales son subdivididas en sub-características. Las sub-características pueden ser medidas con métricas internas o externas. El modelo de calidad para calidad externa e interna del estándar ISO 9126-1 está dividido en seis características (funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad), cada una de las cuales presenta sus sub-características, tal como lo indica la figura 18**.**

A continuación se describe brevemente las características del modelo de calidad para calidad externa e interna, ISO/IEC 9126-1 (2001).

- **Funcionalidad:** Se define como un conjunto de atributos que atañen a la existencia de un conjunto de funciones y sus propiedades específicas. Estas funciones son las que satisfacen las necesidades implícitas y establecidas.
- **Usabilidad.** Capacidad del producto software de ser entendido, aprendido, usado y atraer al usuario, cuando es utilizado bajo ciertas condiciones específicas.
- **Confiabilidad**. Conjunto de atributos que atañen a la capacidad del software para mantener su nivel de prestación bajo condiciones establecidas durante un tiempo establecido.
- **Eficiencia.** Capacidad del producto software para proporcionar un rendimiento apropiado relacionado con el total de recursos utilizados bajo condiciones establecidas.

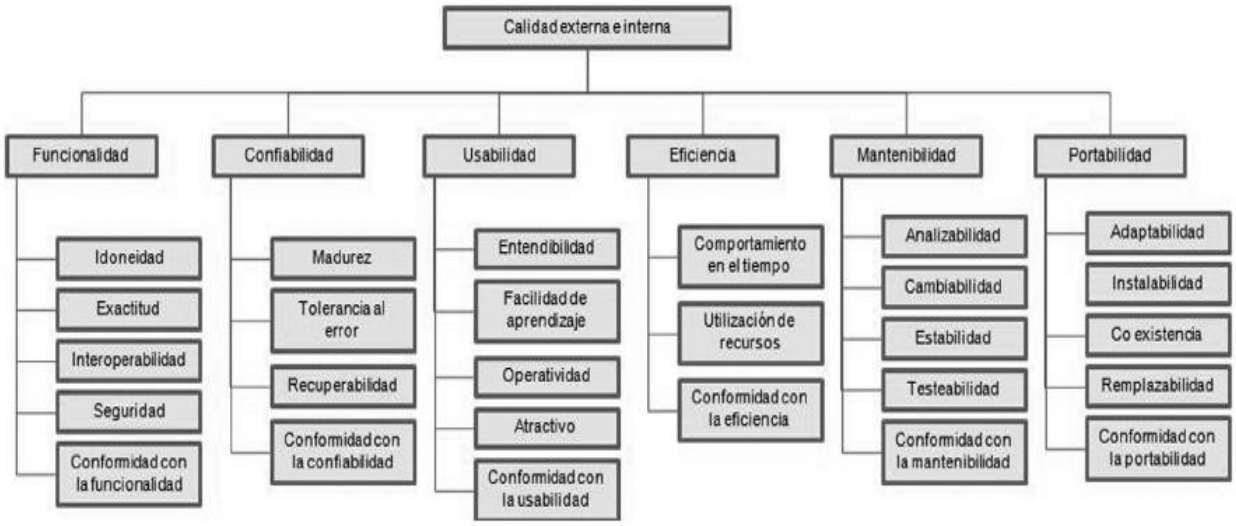

Figura 18. Modelo de calidad para calidad externa e interna del Estándar ISO/IEC 9126
Fuente: Adaptado de ISO/IEC 9126-1 (2001).

- **Mantenibilidad**. Capacidad del producto software para ser modificado.
- **Portabilidad**. Capacidad del producto software para ser transferido de un entorno a otro. El entorno se interpreta tanto a nivel software y hardware, como aquel entorno relacionado con la organización.

De este modo los factores de calidad presentados, tanto los definidos por McCall como los de la norma ISO 9126-1, proporcionan una base valiosa y una lista de validación excelente para evaluar la calidad de un sistema.

Estándar ISO/IEC 25010

La nueva norma ISO/IEC 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software SQuaRE (Software Product Quality Requirements and Evaluation) (ISO/IEC 25000,2005). Estas constituyen una serie de normas basadas en la ISO 9126 (calidad del Producto) y en la ISO 14598 (Evaluación del Software), y su objetivo principal es guiar el desarrollo de los productos de software con la

especificación y evaluación de requisitos de calidad. Establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación. La serie SQuaRE tiene las siguientes divisiones:

- ISO/IEC 2500n. División de dirección de calidad.
- ISO/IEC 2501n. División del modelo de calidad.
- ISO/IEC 2502n. División de medida de calidad.
- ISO/IEC 2503n. División de requisitos de calidad.
- ISO/IEC 2504n. División de evaluación de calidad.

De este modo ISO/IEC 25010, es el estándar que conforma la división del modelo de calidad, presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso, las cuales se detallan a continuación:

- Vista interna: esta vista se ocupa de las propiedades del software como: el tamaño, la complejidad o la conformidad con las normas de orientación a objetos.
- Vista externa: vista que analiza el comportamiento del software en producción y estudia sus atributos, por ejemplo: el rendimiento de un software en una máquina determinada, el uso de memoria de un programa o el tiempo de funcionamiento entre fallos.
- Vista en uso: mide la productividad y efectividad del usuario final al utilizar el software.

El estándar ISO/IEC 25010 describe un modelo bipartito para la calidad del producto de software:

a) Calidad interna y calidad externa.

b) Calidad en el uso.

El modelo de calidad para calidad externa e interna del estándar ISO/IEC 25010 define ocho características de calidad (funcionalidad, seguridad, interoperabilidad,

confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad) cada una de las cuales presenta sus sub-características, tal como lo indica la siguiente figura:

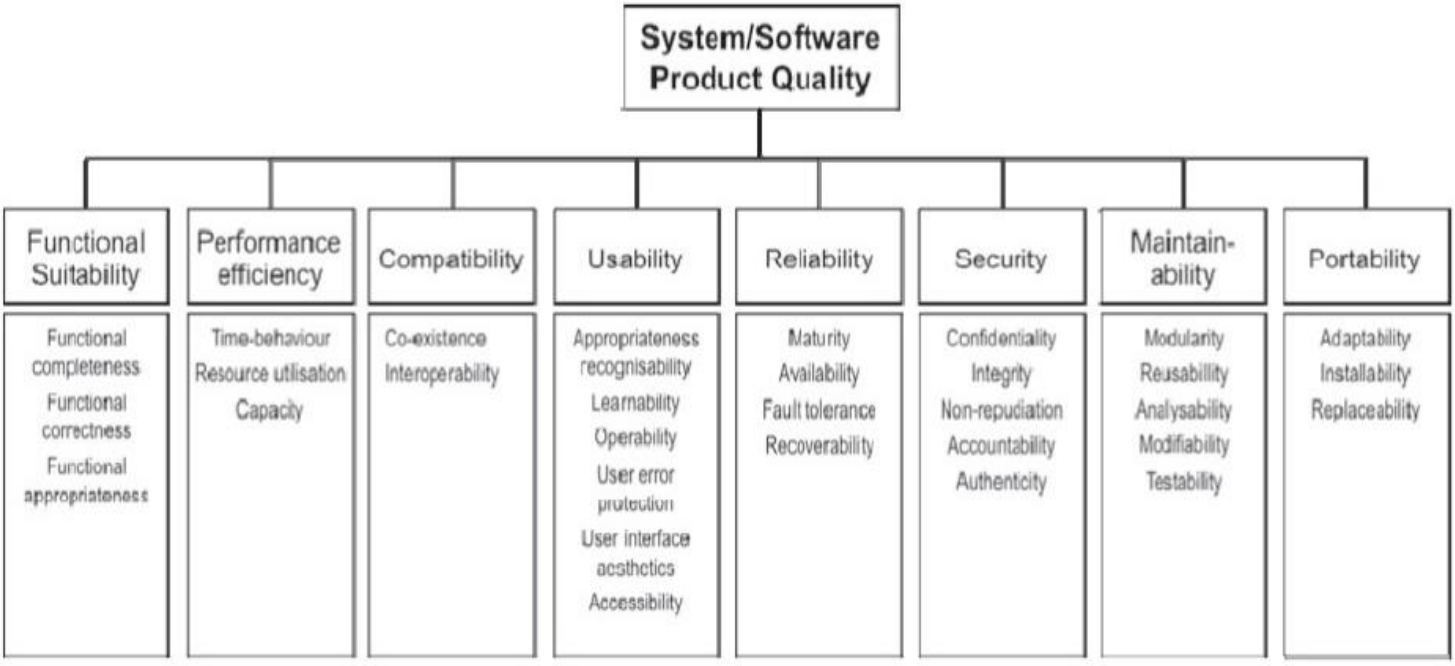

Figura 19. Modelo de calidad para calidad externa e interna del Estándar ISO/IEC 25010
Fuente: Adaptado de ISO/IEC 25010 (2010)

Como se puede observar en la figura anterior las modificaciones de este estándar con respecto al ISO/IEC 9126-1 son las siguientes: **Seguridad**: ha sido añadida como una característica con sub-características asociadas, en lugar de ser una sub-característica de Funcionalidad. **Compatibilidad**: ha sido añadida como una característica con sub-características propias que no estaban en el anterior modelo Se han añadido algunas sub-características que no estaban en el modelo anterior tales como: completitud funcional, disponibilidad, protección contra errores del usuario, modularidad y reusabilidad. Además los nombres de las características son más específicos para evitar la confusión con otros significados más generales.

Fundamentos Metodológicos

A continuación se describirán las disciplinas que guiarán los pasos a seguir para el diseño y construcción del método propuesto en la presente investigación.

Método de Investigación para la Ingeniería de Software

La distinta naturaleza del saber de las ingenierías, con respecto al saber de las ciencias empíricas y formales, hace que los métodos de investigación tradicionales no sean siempre directamente aplicables a la investigación en el ámbito de la ingeniería del software. Es por ello que según la naturaleza del saber de las ingenierías se plantea la necesidad de otro método de investigación que se adapte al saber de la ingeniería del software, de este modo Marcos y Marcos (1998) propusieron un método para la investigación en Ingeniería del Software. Dicho método se basa en el hipotético-deductivo propuesto por Bunge (1983) y se compone de un conjunto de pasos genéricos, mostrados en la figura 20, de forma que pueda aplicarse, con ciertas modificaciones, a la investigación en cualquier disciplina. Marcos y Marcos describen estos pasos o etapas de la siguiente manera:

- **Documentación:** como se observa en la figura 20, la documentación se realiza en todas la etapas del método de investigación. Es importante realizar una documentación profunda acerca del problema a resolver y del método, técnica o proceso para su resolución y validación.

- **Etapa 1, Determinación del Problema:** esta etapa consiste en identificar el cuerpo del conocimiento y una serie de problemas o puntos de mejora en el estado actual del mismo, a partir de ese análisis exhaustivo se determina el problema al cual se quiere dar respuesta.

- **Etapa 2, Creación de la Hipótesis:** una vez definido y delimitado el problema se formula la hipótesis y se definen un conjunto de objetivos que se deben cumplir

para alcanzar la solución que verifique la hipótesis planteada. En los métodos tradicionales de investigación científica la hipótesis se formula en términos causales (sí ocurre A entonces ocurre B). Estas hipótesis son conjeturas de hechos que el método científico deberá contrastar y verificar. Sin embargo, la hipótesis en una investigación en ingeniería del software no responde, por lo general, a un planteamiento de causa-efecto, debido a que el objeto de estudio en este tipo de ciencia es la construcción de nuevos objetos (modelos, técnicas , métodos), la innovación, que, por no existir, no son susceptible de experimentación .

Por tanto, la hipótesis en ciencias de la Ingeniería de Software se formulará como la descripción del nuevo objeto que se desea construir, el proceso o el producto a generar.

- **Etapa 3, Definición del Método de Trabajo:** esta etapa consiste en la definición del método de investigación (método de trabajo) para la resolución y validación del problema. Los autores Marcos y Marcos (1998), recomiendan definir el método a usar, ya que la naturaleza de cada investigación tiene sus propias características y por lo tanto, no sería conveniente aplicar un único método universal de resolución de problemas. Además, debe existir una retroalimentación entre la fase de resolución y validación, y la fase de definición del método de trabajo (tal como lo indica la figura 20), ya que éste se va refinando a medida que se avanza en la resolución del problema. Así, la definición del método de trabajo no concluye hasta que finaliza la etapa de resolución y validación.

- **Etapa 4, Resolución, Validación y Verificación:** en esta etapa se deben especificar las actividades y procesos a realizar según el método definido en la fase anterior para la resolución, validación y verificación del problema. Se presentan juntas las etapas de resolución y validación ya que existe una realimentación entre ellas, de tal modo que el proceso completo de resolución incluye al de validación y verificación.

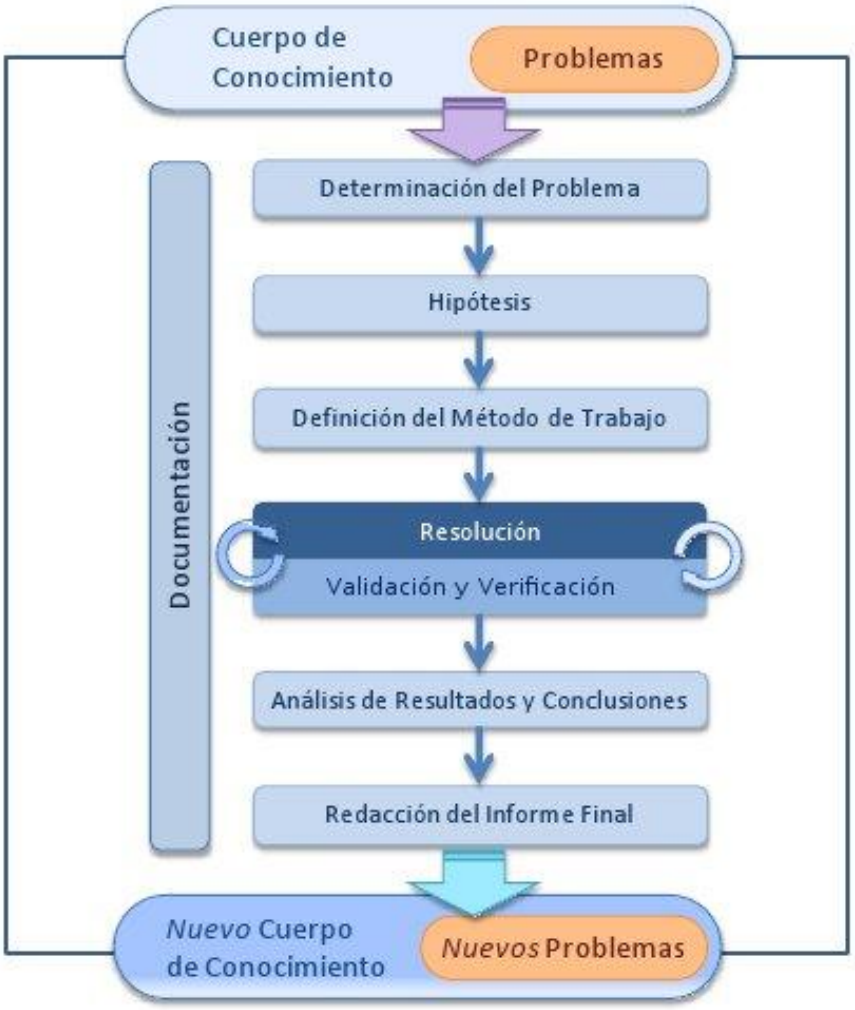

Figura 20. Método Genérico de Investigación propuesto por Marcos y Marcos. Fuente: Adaptado de Marcos y Marcos (1998).

- **Etapa 5, Análisis de Resultados y Elaboración de Conclusiones:** se trata de contrastar la hipótesis planteada al comienzo de la investigación con los resultados obtenidos de ésta. Se deberá comprobar hasta qué punto se han cumplido los objetivos y en qué medida se ha resuelto el problema. En esta fase es muy importante delimitar los aspectos que no se han podido resolver y otros nuevos problemas que hayan surgido como consecuencia de la investigación y que pasarán a ser puntos de partida de nuevas investigaciones.

- **Etapa 6, Redacción del Informe Final**: consiste en la redacción del informe en el que se expone, paso a paso, la investigación realizada. En él se detallará: la hipótesis de trabajo, la justificación del mismo, el método de investigación empleado, el proceso de la investigación, las conclusiones obtenidas, la bibliografía consultada y cualquier otro dato que se considere de relevancia para la comprensión y evaluación del trabajo realizado así como para futuros investigadores que deseen continuar el trabajo emprendido.

Ingeniería de Métodos

Dentro de la disciplina de la Ingeniería de Software, se puede definir método como una serie de pasos a seguir para llevar a cabo una determinada actividad enmarcada dentro de las etapas del ciclo de vida de desarrollo de software (Falkenberg, Hesse y Linggreen, 1996). Los ingenieros de software han creado diferentes métodos y modelos que guían técnicamente la construcción del software, según Sommerville (2005) "un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable"(p. 10).

De este modo, la disciplina que se utiliza para la creación de métodos es Ingeniería de Métodos, la cual según Rolland (1997) "representa el esfuerzo de mejorar la utilidad de los métodos de desarrollo de sistemas, mediante la creación de un marco de adaptación donde los métodos se crean para que coincidan con situaciones concretas de la organización" (p. 1).

Según Brinkkemper (1996), Ingeniería de Métodos, es la disciplina para construir nuevos métodos a partir de métodos ya existentes. Está enfocado al diseño, construcción y evaluación de métodos, técnicas y herramientas de soporte para el desarrollo de sistemas de software.

En este orden de ideas, un tipo de Ingeniería de Métodos es la Ingeniería de

Métodos Situacional, cuya finalidad es la construcción de métodos específicos para un proyecto dado, y basado en situaciones de desarrollo (Sellers y Ralyté, 2010). La Ingeniería de Métodos Situacional propone estrategias para la selección de métodos, llamados bloques de métodos, para crear uno nuevo basado en las características y circunstancias de desarrollo. Así según, Vlaanderen y otros (2008), las ventajas principales de aplicar la Ingeniería de Métodos Situacional son:

- Construir o crear un método base que permita la definición de métodos para un dominio específico, reutilizando bloques validados.
- Adaptar de forma sencilla, métodos basados en las necesidades de un proyecto especifico, con sus restricciones y características
- Proveer las técnicas basadas en la reutilización de bloques ya existentes, en la construcción de nuevos métodos, más flexibles y mejor adaptados a una determinada situación.

En la literatura se utilizan distintos términos para hacer referencia a los bloques de métodos que son el núcleo de los enfoques para la creación de métodos en la Ingeniería de Métodos Situacional, según Brinkkemper (1996) un fragmento de método se caracteriza por tener exactamente un objetivo final que puede ser obtenido por una o más actividades que son realizadas una o más técnicas incluidas en ese fragmento y que pueden generar uno o más artefactos. Así, según Henderson-Sellers y Ralyté (2010), con respecto a la construcción de métodos, los enfoques más utilizados están resumidos en la siguiente tabla:

Tabla 9. Enfoques para la Creación de Métodos en la Ingeniería de Métodos Situacional.

Enfoque	Descripción
Basado en Ensamblaje	Un método nuevo se construye a partir de fragmentos de métodos ya existentes. Los fragmentos pueden ser categorizados en fragmentos de producto y fragmentos de procesos. Este enfoque sigue la estrategia de reutilización
Basado en la Extensión	El método requerido es creado a partir de un método ya existente el cual se extiende haciendo uso de patrones y rellenando las faltas del método original.
Basado en Paradigma	Este enfoque se fundamenta en el concepto que un nuevo método se obtiene a partir de una abstracción de un método existente. Se le llama también enfoque basado en la evolución.

Autor: Adaptado de Henderson-Sellers y Ralyté (2010).

De estos tres (3) enfoques, para la presente investigación se usara el enfoque basado en ensamblaje para construir el método propuesto. El enfoque de la ingeniería de métodos situacional descrita en la mayoría de la literatura es muy claro, Brinkkemper (1996) reconoce los siguientes pasos para la construcción de un nuevo método:

- Caracterización del proyecto
- Selección de fragmentos de método (que se almacenan en un método base) y
- Ensamblaje de los fragmentos de método. La experiencia adquirida en este proceso es una nueva entrada para el método base.

Así mismo, Saeki (2003) afirma que la forma más sencilla de construir un nuevo método es primero colocar los fragmentos de método significativos en un método base, luego se seleccionan fragmentos útiles de este método base, y finalmente adaptarlos e integrarlos en un nuevo método. Por su parte Ralyté, Deneckere y Rolland (2003) han desarrollado el modelo basado en ensamblaje de la ingeniería de

métodos situacional. Este modelo describe tres pasos para desarrollar un método nuevo:

- Especificar los requisitos del método
- Seleccionar fragmentos de método y
- Ensamblar los fragmentos de método.

En este modelo, o bien se asume que el método base con los fragmentos de métodos ya existe, o que los métodos que han de ser almacenados en el método base ya han sido seleccionados. Sin embargo para el caso de los métodos a construir de una información relativamente nueva o cuyo método base no exista, el método base debe ser creado primero. Así, para estos casos las siguientes etapas son propuestas por Weerd y otros (2006), para el enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional:

1. Analizar las situaciones de aplicación y determinar las necesidades.
2. Seleccionar los métodos candidatos que cumplan con uno o más aspectos de las necesidades identificadas.
3. Analizar métodos candidatos y almacenar los fragmentos de métodos relevantes en el método base.
4. Seleccionar fragmentos de método útiles y ensamblarlos en un nuevo método, y utilizar una planificación del desarrollo para obtener métodos situacionales.

Los pasos tercero y cuarto están soportados por una técnica de meta-modelado, especialmente desarrollada para propósitos de la ingeniería de métodos definida por Weerd y Brinkkemper (2009). Esta técnica de meta-modelado es utilizada en el análisis, el almacenamiento, la selección y el ensamblaje de los fragmentos de método, está basada en UML, y representa los fragmentos de un método en un diagrama llamado Process Deliverable Diagram, PDD[15]. Un PDD consiste en dos diagramas

[15] PDD: Diagrama de Procesos y Productos, http://www.igi-global.com/chapter/meta-modeling-situational-analysis-design/21060

integrados, tal como se muestra en la figura 21, la vista de procesos en el lado izquierdo del diagrama se basa en un diagrama de actividad UML, y la vista de entregables en el lado derecho del diagrama se basa en un diagrama de clases UML.

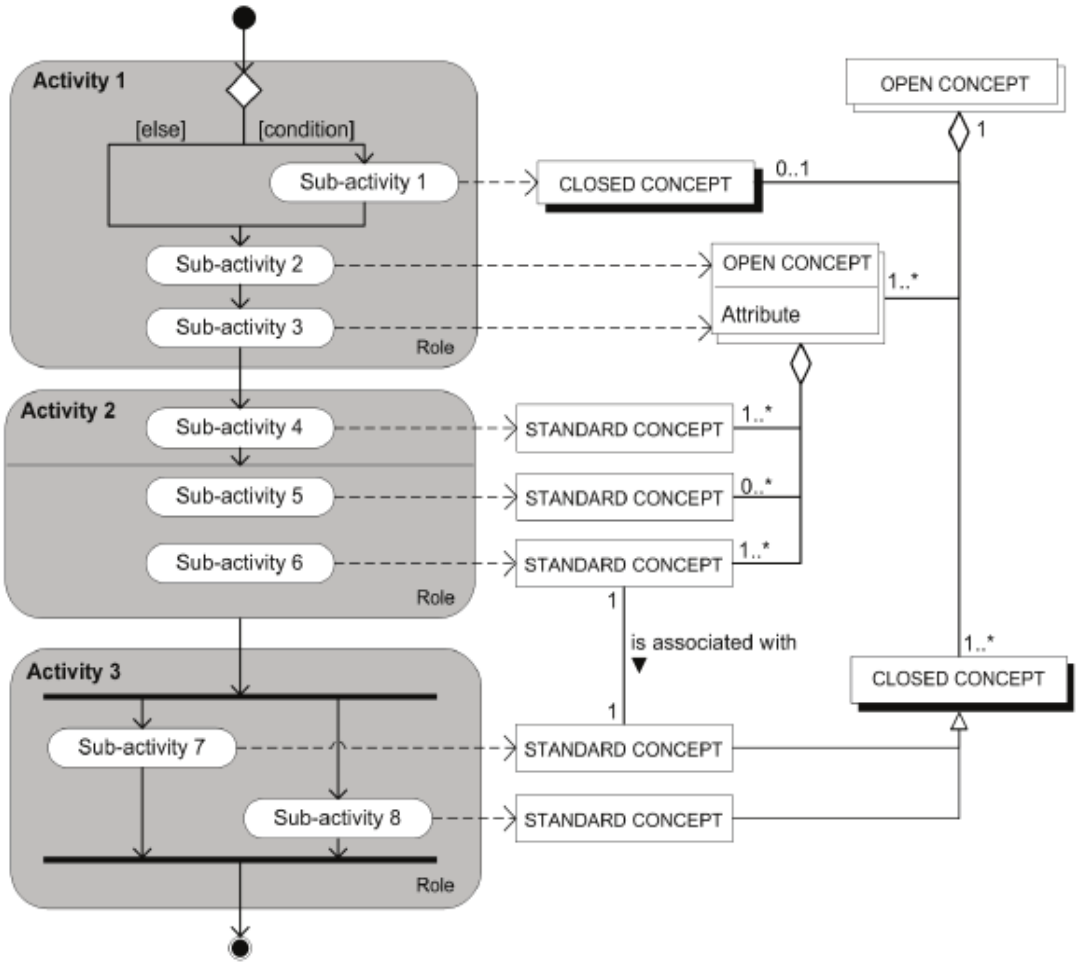

Figura 21 Ejemplo de Process Deliverable Diagram (PDD)
Fuente: Weerd y Brinkkemper (2009).

Las vistas procesos/entregables del diagrama constituyen los diferentes tipos de actividades/conceptos. Estas actividades/conceptos son representadas en la figura 22 y detalladas a continuación:

- Actividad/concepto estándar: una actividad estándar / concepto no contiene más (sub) actividades o conceptos

- Actividad/concepto abierto: una actividad/concepto abierto contiene más (sub) actividades o conceptos.
- Actividad/concepto cerrado: una actividad/concepto cuya (sub) actividades y conceptos no son elaborados ya que no es relevante en el contexto específico.

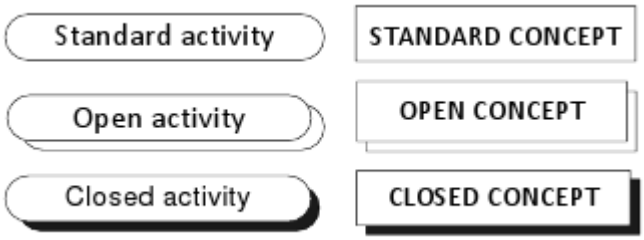

Figura 22. Tipos de Actividades y Conceptos
Fuente: Weerd y Brinkkemper (2009).

Para construir el método propuesto en la presente investigación se usó el enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional definido por Weerd y Brinkkemper (2009).

CAPÍTULO III

MARCO METODOLÓGICO

Toda investigación científica requiere de una metodología previa que le indique al investigador, los pasos necesarios para lograr los objetivos propuestos en la investigación. Esta etapa de la investigación consiste en explicar los aspectos metodológicos que se emplearán para la consecución de los objetivos planteados en la presente investigación.

Naturaleza de la Investigación

La presente investigación se desarrolla según el enfoque de investigación tecnológica, la cual según García-Córdoba (2007), es aquella que tiene como fin obtener un conocimiento para lograr modificar la realidad en estudio, vinculando la investigación y la transformación. La investigación tecnológica, denominada también investigar, idear e innovar, persigue un conocimiento práctico, que sea más un conjunto de instrucciones a seguir para transformar el objeto, que explicaciones teóricas respecto a las cualidades del mismo, (ver figura 23).

García-Cordoba afirma que las perspectivas metodológicas de la investigación en lo social ocurren de dos (2) maneras, estas son la cuantitativa o la cualitativa, pero si se parte de que existe la necesidad de transformar, entonces ni lo cuantitativo ni lo cualitativo es suficiente, es necesario lo tecnológico. Por tanto como la presente investigación está orientada en diseñar un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad, se puede concebir mediante el enfoque de

investigación tecnológica.

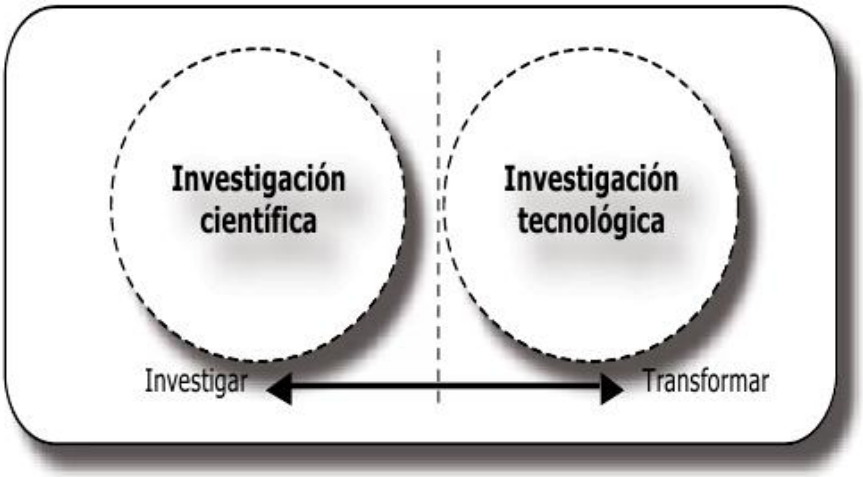

Figura 23. Localización de la Investigación Científica y la Investigación Tecnológica en el continuo Investigar-Transformar.
Fuente: García-Córdoba (2007).

De esta forma una vez definido el enfoque tecnológico de la presente investigación se puede decir que el tipo de estudio en el cual se enmarca este trabajo es bajo la modalidad de Proyecto Especial, según el Manual de Trabajos de Grado de Especialización, Maestría y Tesis Doctorales de la Universidad Pedagógica Experimental Libertador, UPEL (2010), se define de la siguiente manera:

> Trabajos que lleven a creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural. Se incluyen en esta categoría los trabajos de elaboración de libros de texto y de materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general. (p. 22).

Diseño de la Investigación

El diseño de la investigación se refiere al plan o estrategia concebida para responder a las preguntas de investigación, el diseño señala al investigador lo que debe hacer para alcanzar sus objetivos de estudio, contestar las interrogantes que se ha planteado

y obtener la información que desea (Hernández, Fernández y Baptista, 2006).

Así, debido a las diferencias que radican en la naturaleza de las Ingenierías, incluida la Ingeniería del Software, respecto de otras disciplinas empíricas y formales, dificultan la aplicación directa de los métodos de investigación tradicionales. Teniendo en cuenta esta problemática, en 1998, Marcos y Marcos, propusieron un nuevo método para la investigación en la Ingeniería del Software (ver figura 20). Este método ha sido referenciado como diseño metodológico a seguir para el desarrollo de trabajos de grado de maestría y doctorales, así como artículos científicos en los últimos años, en diferentes universidades, siendo alguno de estos: Jimenez (2012), Halim Abang y Deris (2011), López (2011), Valdez (2011) y Santos (2010).

Por lo anteriormente expuesto, el método de investigación que se seguirá en el presente trabajo se basará en el método propuesto por Marcos y Marcos (1998), el cual ha sido previamente definido describiendo cada una de sus etapas en la sección Bases Teóricas.

Procedimiento de la Investigación

Según el alcance definido para este estudio y de acuerdo al método de investigación a seguir para su desarrollo, propuesto por Marcos y Marcos (1998), de las etapas que lo conforman (ver figura 20), se abarca las fases de Resolución, Validación y Verificación, quedando el resto de las etapas para trabajos futuros. De esta forma cada una de las etapas a seguir para llevar a cabo la investigación se describen a continuación:

Documentación

Esta etapa se basa en la revisión bibliográfica documental, la cual consiste en recopilar, identificar, evaluar e interpretar toda la información relativa a la investigación a través de la revisión bibliográfica y consulta en bases de datos científicas. Estas fuentes permiten sustentar científicamente los conceptos, términos, características, teorías, fundamentos a presentar en la investigación, para obtener así,

una mejor compresión de los objetivos propuestos. Esta etapa se debe llevar a cabo a lo largo de todas las etapas del método de investigación**.**

Etapa 1. Determinación del Problema

Esta etapa consiste en identificar el cuerpo del conocimiento y una serie de problemas o puntos de mejora en el estado actual del mismo, a partir de ese análisis exhaustivo se determina el problema al cual se quiere dar respuesta (Marcos y Marcos, 2008). Para el caso del presente estudio el problema que se desea resolver con esta investigación, ha sido planteado detalladamente en el Capítulo I, en la sección Planteamiento del Problema.

Etapa 2. Creación de la Hipótesis

Según Marcos y Marcos, la hipótesis se formula como la descripción del nuevo objeto que se desea construir, el proceso o el producto a generar. Para el caso de esta investigación, la hipótesis es: el diseño de un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad.

Etapa 3. Definición del Método de Trabajo

Esta etapa consiste en la definición del método de trabajo para la resolución del problema. De esta forma, la presente investigación tiene como objetivo general diseñar un método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad, es por ello que el estudio tomará como base la Ingeniería de Métodos Situacional para construir el método propuesto.

Como ya se mencionó anteriormente la Ingeniería de Métodos Situacional, es un tipo de Ingeniería de Métodos cuya finalidad es la construcción de métodos específicos para un proyecto dado y basado en situaciones de desarrollo. (Brinkkemper, 1996). Así con respecto a la construcción de métodos, de los enfoques

más utilizados según Henderson-Sellers y Ralyté (2010) (ver tabla 9), para efectos de esta investigación se tomó el enfoque basado en ensamblaje, ya que el método propuesto se construye a partir de fragmentos de métodos ya existentes.

De esta forma el método de trabajo para la construcción del método propuesto, partiendo del enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional, consiste fundamentalmente en analizar los métodos actuales que den apoyo al proceso de validación de requisitos funcionales de software, patrones RIA y factores de calidad que permitan evaluar la calidad del software, reflexionar acerca de ellos determinando sus ventajas y limitaciones, para así tomar fragmentos de cada uno de ellos para construir un nuevo método que cumpla con el objetivo planteado.

Etapa 4. Resolución, Validación y Verificación

En esta etapa se deben especificar los pasos y procesos a seguir, según el método definido en la fase anterior, para la resolución, validación y verificación del problema. Según el alcance de la investigación definido en el presente trabajo la resolución del problema, llega hasta la construcción del método propuesto, donde finalmente se ejemplifica el uso del método a través de un caso de estudio para validar cada una de las actividades y productos que lo componen, quedando el paso correspondiente a la verificación para trabajos futuros. De esta forma las actividades a seguir en esta etapa, las cuales ayudan a cumplir los objetivos planteados en la presente investigación, son las siguientes:

- Análisis: permite describir el estado actual del objeto en estudio. Se analizan las tendencias actuales en cuanto a métodos, procesos y/o modelos validados, probados y empleados en el campo de la Ingeniería del Software, que den apoyo al proceso de validación de requisitos, patrones RIA y factores de calidad para evaluar la calidad del software por parte del usuario.
- Diseño: de acuerdo al resultado obtenido en la actividad de análisis, una vez identificados y estudiado los métodos, procesos y/o modelos actuales que den apoyo al problema planteado, estos servirán como base para el diseño del método

propuesto, ya que el mismo se construirá a partir del enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional, donde se tomaran fragmentos de cada uno de los métodos, procesos y/o modelo estudiados, determinando sus ventajas y limitaciones, para así tomar características de cada uno de ellos. De este modo a partir de ese análisis se podrán:

- Determinar el flujo de trabajo que debe seguir el método propuesto. Se usó el enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional, propuesto por Weerd y otros (2006), para determinar el flujo de trabajo del método propuesto. A continuación se detallan los pasos a seguir, según este enfoque:
  - ✓ Analizar las situaciones de aplicación y determinar las necesidades.
  - ✓ Seleccionar los métodos candidatos que cumplan con uno o más aspectos de las necesidades identificadas.
  - ✓ Analizar métodos candidatos y almacenar los fragmentos de métodos relevantes en el método base.
  - ✓ Seleccionar fragmentos de método útiles y ensamblarlos en un nuevo método.

    Los dos últimos pasos de este enfoque están soportados por la técnica de meta-modelado de Weerd y Brinkkemper (2009), que permite representar un método y sus fragmentos independientes, en un PDD (Process Deliverable Diagram).
- Definir las métricas (parámetros de medición) a utilizar por cada uno de los factores de calidad a ser aplicados en la evaluación de la calidad del prototipo de interfaces de usuarios basado en patrones RIA.
  - ✓ Seleccionar las características de calidad del modelo ISO/IEC 25010, tomando como base las características de calidad seleccionadas como resultado de la combinación de los modelos McCall e ISO 9126-1, del estudio realizado por Macías y Gómez (2010).

- ✓ Definir las métricas por cada factor de calidad a utilizar para la evaluación de la calidad por parte de los usuarios, del prototipo de interfaces de usuario basado en patrones RIA utilizado en el método de validación de requisitos funcionales propuesto.
- ✓ Asociar los patrones RIA a los parámetros de medición según los factores de calidad seleccionados para la evaluación de calidad del prototipo que apoyará al proceso de validación de requisitos funcionales de software.

- Construcción: una vez determinado el flujo de trabajo a seguir en el método, y definidas las métricas a usar por cada uno de los factores de calidad a ser aplicados, se procederá al diseño final del método, donde se definirán con detalle cada uno de los pasos a seguir indicando los productos (artefactos) de entrada y salida para cada paso.
- Validar el método propuesto ejemplificando su uso a partir de un caso de estudio.

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

Luego de haber examinado los antecedentes y el marco teórico relacionado con esta investigación, en este capítulo se presenta la propuesta del método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad, el cual recibe el nombre de **VAREME** (*Validation Requirements Method*). El desarrollo de la propuesta está enmarcado dentro del paso número cuatro (4) del método de investigación para la Ingeniería de Software de Marcos y Marcos (1998), denominados Resolución, Validación y Verificación, tal como se indico en el capítulo III.

Para ello, la propuesta se inicia con la construcción del método partiendo del enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional, donde tomando como base los tres métodos de validación de requisitos a través de prototipos, presentados en la sección Antecedentes, se construye un nuevo método.

De este modo para cumplir con los objetivos planteados en la presente investigación, es necesario incorporar al método resultante, la definición y uso de los factores de calidad a utilizar en la evaluación de la calidad del prototipo de interfaces de usuarios, así como los patrones RIA que se deben tomar en cuenta para la construcción del prototipo. Finalmente se presenta un caso de estudio donde se ejemplificara el uso del método propuesto.

A continuación se presenta un mapa conceptual de los trabajos de investigación utilizados como base para la construcción del método propuesto, ver la figura 24.

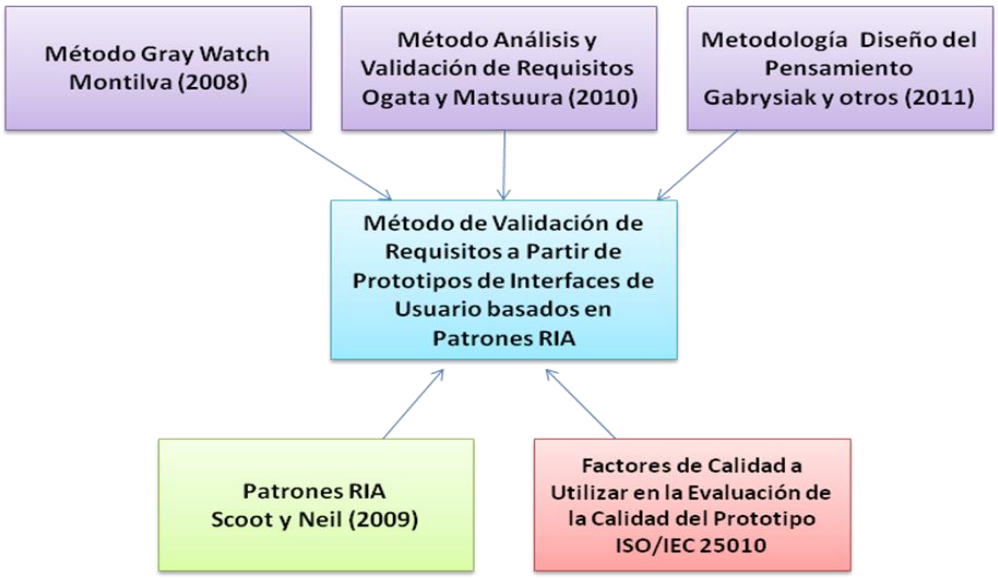

Figura 24. Mapa Conceptual. Trabajos de Investigación tomados como base para la construcción del Método Propuesto.
Fuente: La Autora (2012).

El objetivo del método propuesto, denominado VAREME, es permitir al ingeniero de software validar requisitos funcionales de software a partir de prototipos de interfaces de usuario, usando para ello un enfoque enmarcado en métricas de calidad e incorporando los patrones de interfaz humano-computador Rich *Internet Applications* (RIA). De esta manera se podrá establecer una mejor concordancia entre el sistema y las necesidades del usuario, permitiendo la posibilidad de desarrollar un sistema que satisfaga en mejor forma las necesidades y expectativas de los usuarios y por lo tanto ayude a garantizar el éxito en la Ingeniería de Requisitos.

Estructura de la Propuesta

A continuación se detallan los pasos a seguir para llevar a cabo la propuesta presentada:

1. Definición del flujo de trabajo que debe seguir el método propuesto, VAREME: en el capítulo II se presentaron tres (3) métodos de validación de requisitos de software a

través de prototipos de interfaces de usuario, los cuales sirven como base para el diseño del método VAREME, ya que el mismo se construirá a partir del enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional, donde se toman fragmentos de cada uno de los métodos, procesos y/o modelo estudiados, para así tomar características de cada uno de ellos.

Como se mencionó anteriormente una estrategia específica de la ingeniería de métodos situacional es la basada en ensamblaje, en donde un método nuevo se construye a partir de fragmentos de métodos ya existentes tomados de un método base, este enfoque incluye entre sus pasos el crear un método base en el caso que no exista. Un método base es un repositorio donde los fragmentos de métodos pueden ser almacenados y recuperados para su uso. En este caso para construir el método propuesto en la presente investigación no existe un método base del cual los fragmentos de métodos puedan tomarse para ser reutilizados, es por ello, que se usó el enfoque basado en ensamblaje de Weerd y otros (2006) para construir el método VAREME. A continuación se detallan los pasos a seguir, según el enfoque basado en ensamblaje según Weerd y otros (2006):

a. **Analizar la situación de aplicación e identificar necesidades**: el análisis de la situación y las tendencias actuales en el área de estudio así como la necesidad e importancia del método propuesto han sido descritos en los capítulos I y II de la presente investigación.
b. **Seleccionar métodos candidatos:** en este paso se seleccionan los métodos de validación de requisitos de software a través de prototipos de interfaces de usuario que contengan fragmentos de métodos relevantes. Una vez estudiadas las tendencias actuales en cuanto a métodos, procesos y/o modelos validados, probados y empleados en el campo de la Ingeniería del Software, que dan apoyo al proceso de validación de requisitos, se seleccionaron tres métodos: el Método Gray Watch, de Montilva y otros (2008), el método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de

Usuario para Aplicaciones Web, de Ogata y Matsuura (2010) y la metodología diseño del pensamiento: escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios, de Gabrysiak y otros (2011), los cuales están explicados con detalle en el capítulo II. En lo sucesivo estos métodos serán denominados "métodos candidatos".

c. **Analizar métodos candidatos y almacenar los fragmentos de métodos relevantes en el método base:** en este paso el método base es formado con los fragmentos de métodos relevantes derivados de los métodos candidatos, esto se realiza mediante el análisis de los métodos candidatos para crear los diagramas de procesos y productos (PDD), donde para cada método actividades y sub-actividades son identificadas, resultando en un producto entregable (artefacto), lo cual debe ser representado en el PDD.

Así, para cada método candidato se deber crear un PDD, mediante la aplicación de la técnica de metamodelado según Weerd y Brinkkemper (2009). Los PDD de los métodos candidatos representan todas las actividades y productos entregables de cada fase. Un PDD, tal como se especificó anteriormente, consiste en dos diagramas integrados (ver figura 25), la vista de procesos en el lado izquierdo del diagrama se basa en un diagrama de actividad UML, y la vista de productos en el lado derecho del diagrama se basa en un diagrama de clases UML. De este modo, todos los diagramas que son producidos en este paso conforman el método base, del cual se seleccionan los fragmentos para ensamblar el método propuesto.

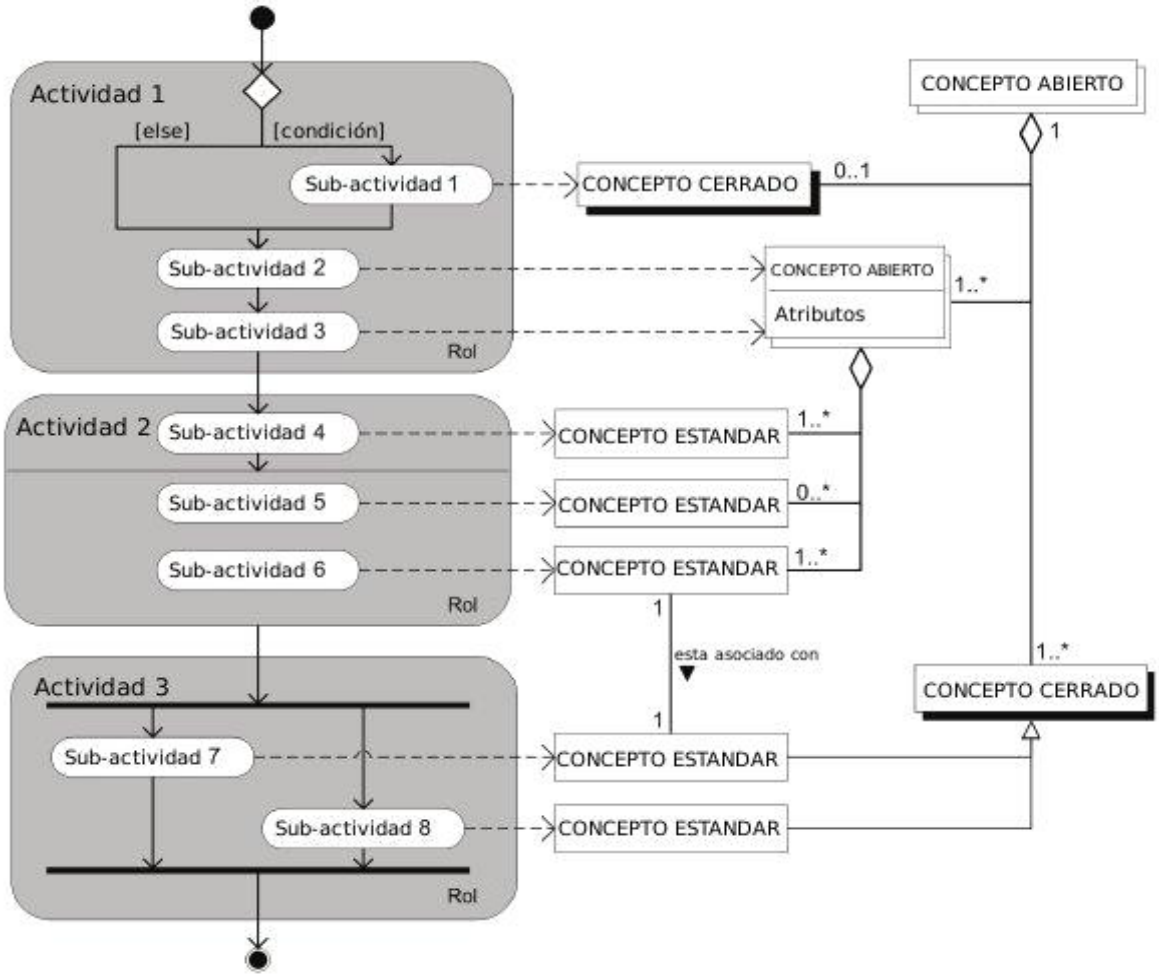

Figura 25. Ejemplo de Diagrama de Procesos y Productos (PDD)
Fuente: Adaptado de Weerd y Brinkkemper (2009).

A continuación se muestran los PDD de los métodos candidatos, cada método está representado como un todo, donde cada fase y actividad puede considerarse como un fragmento de método independiente. Al describir los métodos en su conjunto se mantiene una visión clara del método.

i. PDD del Subproceso de Validación de Requisitos del Método Gray Watch (Montilva y otros, 2008)

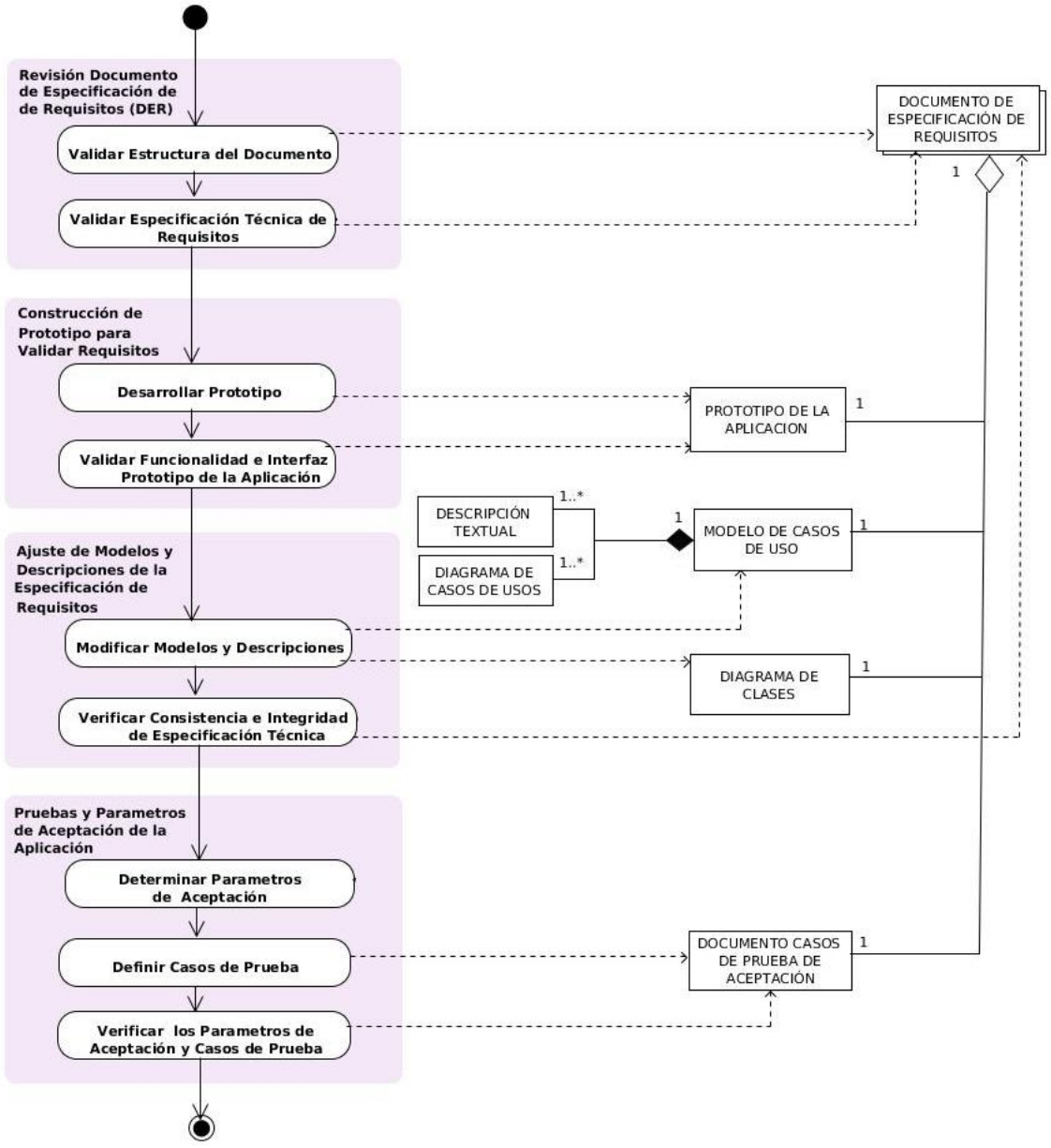

Figura 26. PDD del Subproceso de Validación de Requisitos. Método Gray Watch. Fuente: La Autora (2012)

A continuación se detallan cada una de las actividades y sub-actividades del subproceso de validación de requisitos del método Gray Watch:

Tabla 10. Actividades del Subproceso de Validación de Requisitos. Método Gray Watch

Actividad	Sub-Actividad	Descripción
Revisión Documento de Especificación de Requisitos (DER)	Validar la estructura del documento	Revisión y validación de la estructura y contenido del DOCUMENTO DE ESPECIFICACION DE REQUISITOS, este al cual se le realizan las correcciones pertinentes si lo amerita. Este documento contiene especificaciones técnicas de los modelos de casos de uso y de clases.
	Validar especificación técnica de los requisitos	Revisión y validación de las especificaciones técnicas de los requisitos contenidos en el DOCUMENTO DE ESPECIFICACION DE REQUISITOS: MODELO DE CASOS DE USO y DIAGRAMA DE CLASES, a los cuales se les realizan las correcciones pertinentes si lo amerita.
Construcción de Prototipo para Validar Requisitos	Desarrollar prototipo	Desarrollar un PROTOTIPO DE LA APLICACIÓN que emule la funcionalidad y la interfaz que tendría la aplicación, según MODELO DE CASOS DE USO, DIAGRAMA DE CLASES y el DOCUMENTO DE ESPECIFICACION DE REQUISITOS.
	Validar funcionalidad e interfaz prototipo de la aplicación.	En sesiones con los usuarios finales se validan los requisitos a través del PROTOTIPO DE LA APLICACIÓN, validando funcionalidad y las interfaces de usuario.
Ajuste de Modelos y Descripciones de la Especificación de Requisitos	Modificar modelos y descripciones	Modificar los MODELO DE CASOS DE USO, DIAGRAMA DE CLASES atendiendo a los resultados de la validación del PROTOTIPO DE LA APLICACIÓN.
	Verificar consistencia e integridad de la especificación técnica.	En base a los resultados de la validación de requisitos y los ajustes a los modelos correspondientes, verificar consistencia e integridad de la especificación técnica.
Definición de Pruebas y Parámetros de Aceptación de la Aplicación	Determinar parámetros de aceptación de la aplicación.	Definir parámetros de aceptación a utilizar para lograr un acuerdo con los usuarios finales luego de haber validado los requisitos a través del PROTOTIPO DE LA APLICACIÓN.
	Definir casos de prueba de aceptación.	Definir DOCUMENTO DE CASOS DE PRUEBA DE ACEPTACIÓN de tipo funcional, realizadas directamente por los usuarios del sistema. Este tipo de prueba se centra en la interfaz de usuario y en la funcionalidad del PROTOTIPO DE LA APLICACIÓN.
	Verificar los parámetros de aceptación y los casos de prueba.	Verificar, con los interesados (usuarios finales), los parámetros de aceptación y el DOCUMENTO DE CASOS DE PRUEBA DE ACEPTACIÓN de la aplicación.

Fuente: Adaptado de Montilva y otros (2008)

ii. PDD del Método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web (Ogata y Matsuura, 2010)

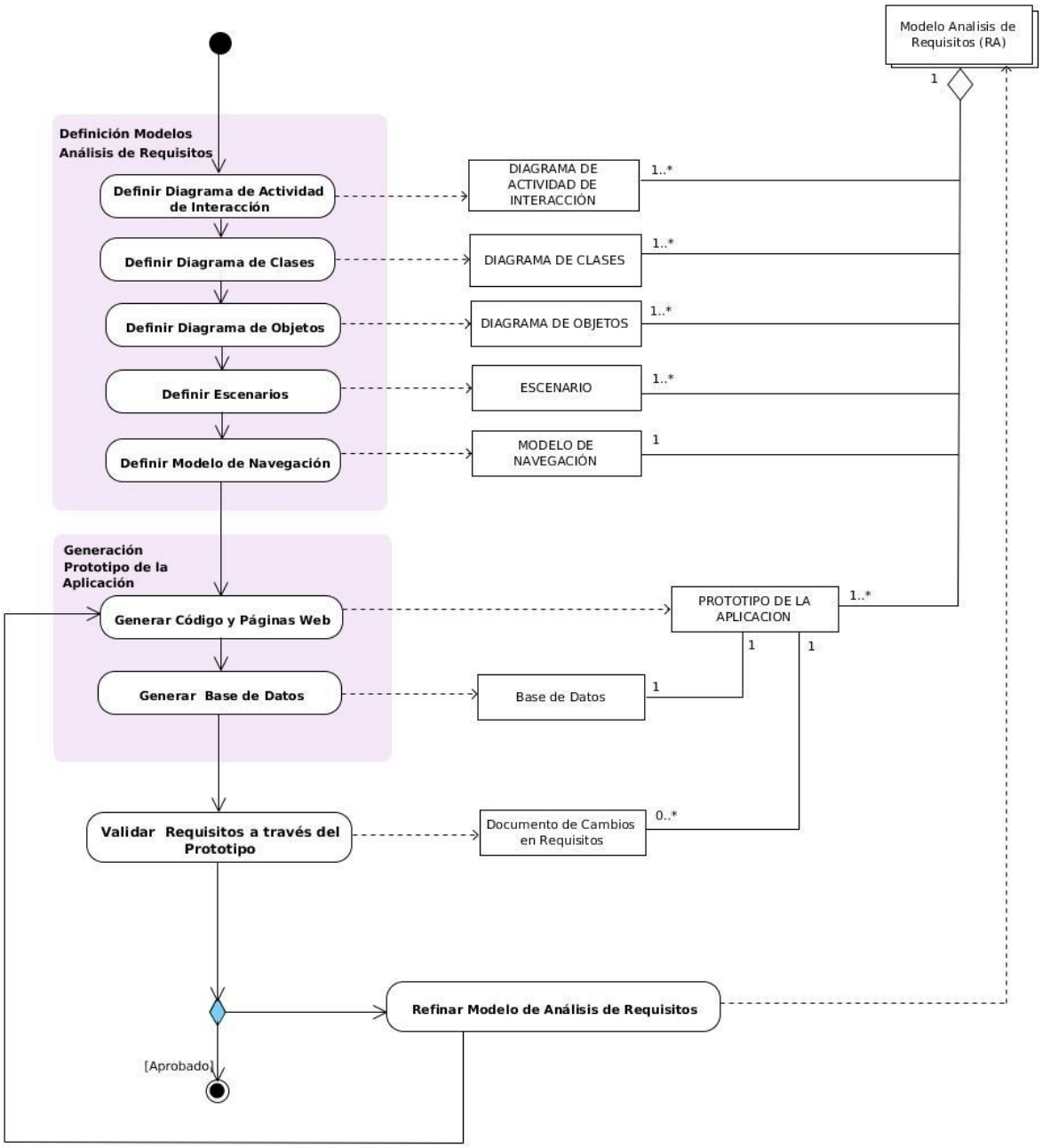

Figura 27. PDD del Método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web.
Fuente: La Autora (2012)

Tabla 11. Actividades del Método de Análisis y Validación de Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web.

Actividad	Sub-Actividad	Descripción
Definición de Modelos de Análisis de Requisitos (Modelo RA)	Definir Diagrama de Actividad de Interacción	Definir los DIAGRAMAS DE ACTIVIDAD DE INTERACCIÓN los cuales definen las interacciones entre los actores y el sistema usando un diagrama de actividad en UML con el mismo concepto de la plantilla de caso de uso.
	Definir Diagrama de Clases	Definir un DIAGRAMA DE CLASES, el cual define estructuras de datos como clases. Los datos visibles para los usuarios, de los elementos importantes para lograr el flujo de trabajo empresarial correctamente.
	Definir Diagrama de Objetos	Los datos concretos es la forma más fácil para los usuarios de entender la información, así que datos concretos se deben utilizar para validar los requisitos efectivamente. Los DIAGRAMAS DE OBJETOS definen las especificaciones de instancias de clases como datos concretos en una situación específica del uso del sistema, en un escenario.
	Definir Escenarios	Los desarrolladores deben garantizar a los usuarios como llevar a cabo correctamente el flujo de trabajo empresarial en una situación específica esperada por los usuarios. Para lograr este propósito, un camino de los flujos de interacción se define como un ESCENARIO.
	Definir Modelo de Navegación	Definir un MODELO DE NAVEGACIÓN el cual define la secuencia a seguir luego de la invocación de un servicio, una acción se corresponde con un servicio. Se define mediante un diagrama de actividad en UML.
Generación Prototipo de Interfaces de Usuario de Aplicaciones Web	Generar Código y Páginas Web	A partir del modelo RA se genera automáticamente el PROTOTIPO DE LA APLICACIÓN, el cual representa la implementación de todos los diagramas del modelo RA. El resultado de este proceso es un conjunto de páginas HTML que conforman un PROTOTIPO funcional. Así, a pesar de definir flujos de interacción detallados de forma técnica, los clientes pueden validar el MODELO RA y por lo tanto los requisitos, a través del PROTOTIPO generado.

	Generar Base de Datos	Las clases definidas en el DIAGRAMA DE CLASES se convierten en tablas que representan la estructura de datos, de esta forma a partir de las clases definidas se genera la Base de datos con sus correspondientes tablas, donde cada clase con sus atributos corresponde a una tabla con sus campos respectivos.
Validar Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web		Los desarrolladores y los usuarios validan el MODELO RA utilizando el PROTOTIPO generado en sesiones de validación, así los usuarios interactúan con el PROTOTIPO para validar los requisitos elicitados. El resultado de este proceso pueden ser problemas los cuales incluyen errores de definición, requisitos omitidos, cambios en los requisitos, entre otros, que se formalizan en un DOCUMENTO DE CAMBIOS DE REQUISITOS. De esta forma es necesario modificar el MODELO RA mediante la corrección de los problemas descubiertos, se genera nuevamente el PROTOTIPO, luego se realizan las sesiones de validación, y se ejecutan estos pasos iterativamente hasta que los usuarios estén satisfechos con el PROTOTIPO DE LA APLICACIÓN generado, lo que implica satisfacción con el MODELO RA y por lo tanto con los requisitos.
Refinar Modelo de Análisis de Requisitos (Modelo RA)		Los desarrolladores modifican el MODELO RA mediante la corrección de los problemas descubiertos en la validación del paso anterior. La salida de este proceso es el modelo RA refinado

Fuente: Adaptado de Ogata y Matsuura (2010)

iii. PDD del Proceso del Proceso de Validación de Requisitos de la Metodología Diseño del Pensamiento: escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios (Gabrysiak y otros, 2011)

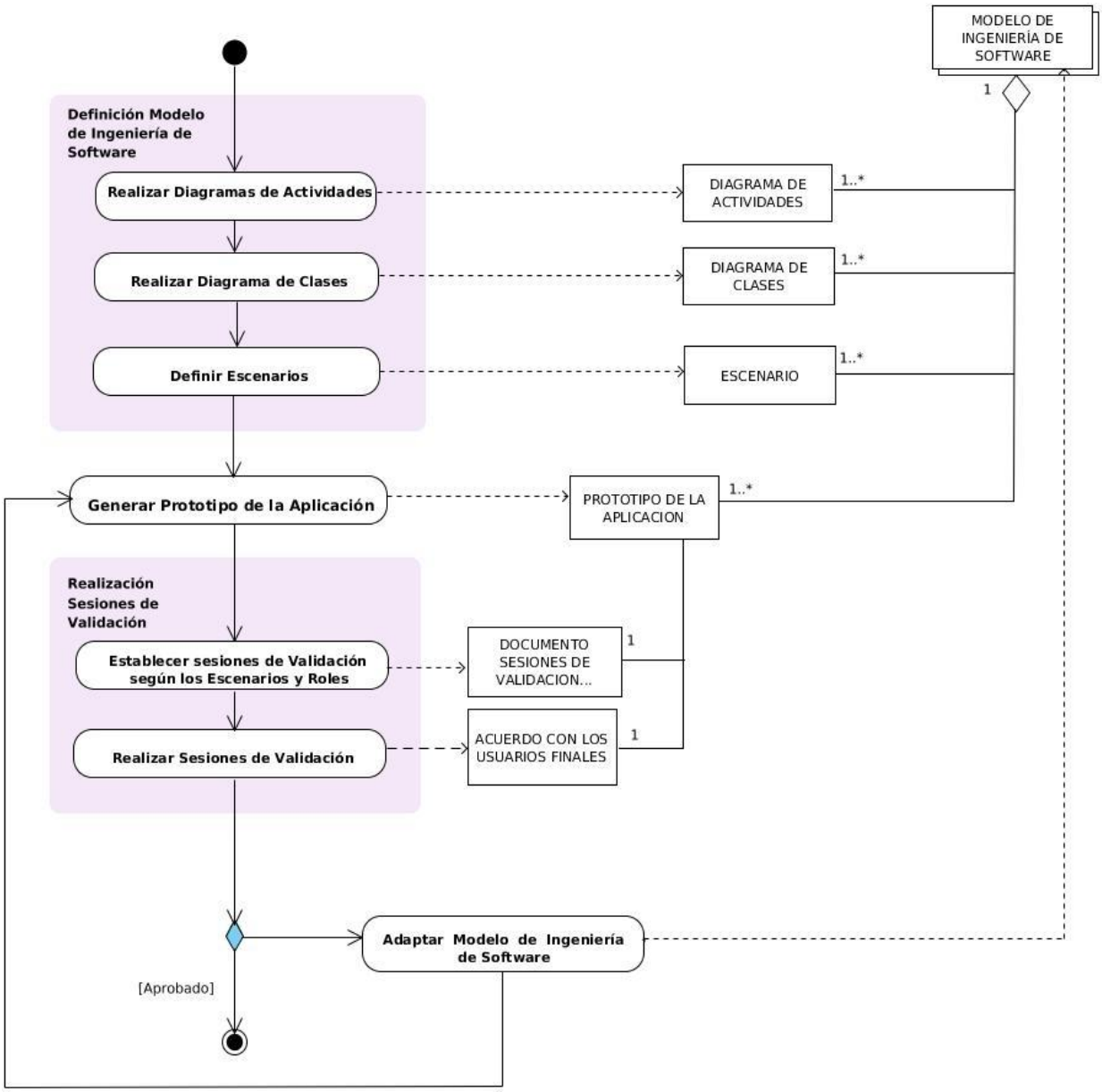

Figura 28. PDD del Proceso de Validación de Requisitos. Metodología Diseño del Pensamiento: escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios. Fuente: La Autora (2012)

A continuación se detallan cada una de las actividades y sub-actividades del subproceso de validación de requisitos de esta metodología:

Tabla 12. Actividades de la Metodología Diseño del Pensamiento: Escenarios basados en prototipos para el diseño de sistemas de software complejos multiusuarios.

Actividad	Sub-Actividad	Descripción
Definición Modelo de Ingeniería de Software	Realizar Diagrama de Actividades	En base a la elicitación de requisitos se identifican las actividades, se establece un preliminar orden causal de esas actividades y se realizan los DIAGRAMA DE ACTIVIDADES.
	Realizar Diagrama de Clases	Se realiza el DIAGRAMA DE CLASES para especificar los conceptos de dominio y las relaciones entre ellos.
	Definir Escenarios	Se definen los ESCENARIOS, donde cada escenario puede ser considerado como un conjunto de diagramas de secuencia donde se relaciona cada interacción con una especificación de comportamiento que define la condición de la interacción.
Construir Prototipo de la Aplicación		Construcción del PROTOTIPO DE LA APLICACIÓN utilizando una herramienta de generación semi-automática, partiendo de los MODELOS DE INGENIERÍA DE SOFTWARE : MODELO DE CASOS DE USO, DIAGRAMA DE ACTIVIDADES y el DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS. El prototipo es una interpretación tangible de dichos modelos.
Realización de Sesiones de Validación	Establecer sesiones de validación según los escenarios y Roles	En base a los escenarios y roles definidos en la fase de definición de modelos de ingeniería de software, se establecen y organizan las SESIONES DE VALIDACIÓN. Partiendo de estos escenarios se debe seleccionar uno o un conjunto de usuarios representativo de los distintos roles definidos, de tal modo que sea posible validar el PROTOTIPO DE LA APLICACIÓN en todos sus modos de utilización.

	Realizar sesiones de validación	Realizar las SESIONES DE VALIDACIÓN, para validar los requisitos a través de la interacción de los usuarios finales con el PROTOTIPO DE LA APLICACIÓN. Según la retroalimentación proporcionada durante la interacción con el PROTOTIPO DE LA APLICACIÓN, los modelos de ingeniería de software deben ser adaptados para incorporar los cambios surgidos, se genera nuevamente el PROTOTIPO DE LA APLICACIÓN en base a estos cambios y se repite el proceso de validación. Esta secuencia puede repetirse sistemáticamente hasta que exista un común entendimiento del dominio del problema, generando un documento de ACUERDO CON LOS USUARIOS FINALES y culminando el proceso de validación.
Adaptar Modelo de Ingeniería de Software		Según la retroalimentación proporcionada durante la las sesiones de validación, los MODELOS DE INGENIERÍA DE SOFTWARE: MODELO DE CASOS DE USO, DIAGRAMA DE ACTIVIDADES y el DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS deben ser adaptados para incorporar los cambios surgidos en estas sesiones.

Fuente: Adaptado de Gabrysiak y otros (2011)

d. **Seleccionar fragmentos de método útiles y ensamblarlos en un nuevo método:** Esta sección consiste en diseñar el método VAREME a partir de la selección de los fragmentos útiles del método base, en correspondencia con la cuarta etapa del enfoque adoptado de Weerd y otros (2006). El nombre de este paso fue modificado, de "Seleccionar fragmentos de método útiles y ensamblarlos en un nuevo método, y utilizar una planificación del desarrollo para obtener métodos situacionales", a usar el nombre "Seleccionar fragmentos de método útiles y ensamblarlos en un nuevo método", se retira el uso de una planificación del desarrollo para obtener métodos situacionales, debido a que el método propuesto no detalla diferentes situaciones, de la misma forma como este paso lo realiza Reijnders y otros (2011), en su propuesta para la construcción de un nuevo método.

Fases del Método VAREME

Al analizar el método base para la selección y adopción de los fragmentos del método se determina que los fragmentos están representados en actividades y en productos entregables, ninguna fase en su conjunto será seleccionada de un determinado método candidato. Por ello el primer paso es definir las fases del método VAREME, partiendo de las diferentes fases de los métodos candidatos, según Reijnders y otros (2011).

Para determinar las fases del método VAREME, los métodos que conforman el método de base se comparan para analizar las similitudes entre sus fases. Las fases que resulten ser similares en intención se agrupan para formar las fases del método VAREME. El resultado de la comparación se representa en la siguiente tabla, de la cual se concluye que las fases entre los diferentes métodos son similares facilitando la identificación de las fases para el método VAREME.

Tabla 13. Comparación Fases de los Métodos pertenecientes al Método Base

	Montilva (2008)	Ogata y Matsuura (2010)	Gabrysiak y otros (2011)
1	Revisión Documento de Especificación de Requisitos (DER)	Definición de Modelos de Análisis de Requisitos (Modelo RA)	Definición Modelo de Ingeniería de Software
2	Construcción de Prototipo para Validar Requisitos	Generación Prototipo de Interfaces de Usuario de Aplicaciones Web	Construir Prototipo de la Aplicación
3		Validar Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web	Realización de Sesiones de Validación
4	Ajuste de Modelos y Descripciones de la Especificación de Requisitos	Refinar Modelo de Análisis de Requisitos (Modelo RA)	Adaptar Modelo de Ingeniería de Software
5	Definición de Pruebas y Parámetros de Aceptación de la Aplicación	No está presente	No está presente

Fuente: La Autora (2012)

Por lo tanto, las fases definidas para el método VAREME son las siguientes:

1. Revisión Documento de Especificación de Requisitos
2. Generación Prototipo de Interfaces de Usuario para Aplicaciones Web
3. Validación de Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web
4. Ajuste Documento de Especificación de Requisitos
5. Definición de Parámetros de Aceptación de la Aplicación

Una vez definidas las fases se definen las actividades para cada una de estas fases, las cuales se seleccionan y se reutilizan de los métodos que conforman el método base. La selección se lleva a cabo mediante la comparación de cada actividad con la situación y las necesidades definidas para el método propuesto. Algunas actividades están fuera del alcance del método propuesto y por esa razón se dejan fuera. La matriz resultante de la comparación de los métodos se representa en la tabla 14, la cual se crea sobre la base de las técnicas de comparación utilizadas por Reijnders y otros (2011), Weerd y otros (2007) y Hong y otros (1993).

En base a esta matriz de comparación un nuevo método es creado, donde todas las actividades comunes entre los tres métodos están incluidos en un solo método, (ver tabla 14).

Tabla 14. Matriz Comparación Métodos

Montilva (2008)	Ogata y Matsuura (2010)	Gabrysiak y otros (2011)
1. Revisión Documento de Especificación de Requisitos		
Validar la estructura del documento		
Validar especificación técnica de los requisitos	Definir Diagrama de Actividad de Interacción	Realizar Diagramas de Actividades
	Definir Diagrama de Clases	Realizar Diagrama de Clases
	Definir Diagrama de Objetos	-
	Definir Escenarios	Definir Escenarios

	Definir Modelo de Navegación	-
2. Generación Prototipo de Interfaces de Usuario de Aplicaciones Web		
Desarrollar prototipo	Generar Código y Páginas Web	Construir Prototipo de la Aplicación
-	Generar Modelo de Datos (Base de Datos Relacional o NoSQL)	-
3. Validación de Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web		
Validar funcionalidad e interfaz prototipo de la aplicación.	Validar Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web	Establecer sesiones de validación según los escenarios y Roles
-	-	Realizar sesiones de validación
4. Ajuste Documento de Especificación de Requisitos		
Modificar modelos y descripciones	Refinar Modelo de Análisis de Requisitos (Modelo RA)	Adaptar Modelo de Ingeniería de Software
Verificar consistencia e integridad de la especificación técnica.	-	-
5. Definición de Parámetros de Aceptación de la Aplicación		
Determinar parámetros de aceptación de la aplicación.	-	-
Definir casos de prueba de aceptación.	-	-
Verificar los parámetros de aceptación y los casos de prueba.	-	-

Fuente: La Autora (2012)

De este modo según la matriz de comparación se desprenden las siguientes actividades para cada una de las fases especificadas, para el método VAREME:

i. **Revisión Documento de Especificación de Requisitos:** según Montilva y otros (2008), en los subprocesos previos al de Validación de Requisitos, se realiza el Documento de Requisitos de la Aplicación, el cual es el producto principal del proceso

de Ingeniería de Requisitos, el cual describe el conjunto de requisitos que establecen los usuarios de la aplicación. Tal como ilustra la figura 29, este documento está dividido en tres partes. La primera de ellas es la Definición de Requisitos, la cual describe los requisitos desde la perspectiva de los usuarios de la aplicación empresarial; está dirigida a los usuarios y demás interesados en la aplicación. No tiene, por lo tanto, un carácter técnico. La segunda es denominada Especificación de Requisitos de la Aplicación (DER) y consta de un conjunto de modelos elaborados usando la notación UML; está dirigida al Grupo de Diseño que tiene a su cargo elaborar el diseño de la aplicación, por consiguiente, tiene un carácter técnico. La tercera parte corresponde al Plan de Gestión de Requisitos de la aplicación, está dirigido al Grupo de Análisis (Gestor de Requisitos) y establece el conjunto de actividades que se deben realizar para llevar a cabo este proceso, durante todo el proceso de desarrollo de la aplicación.

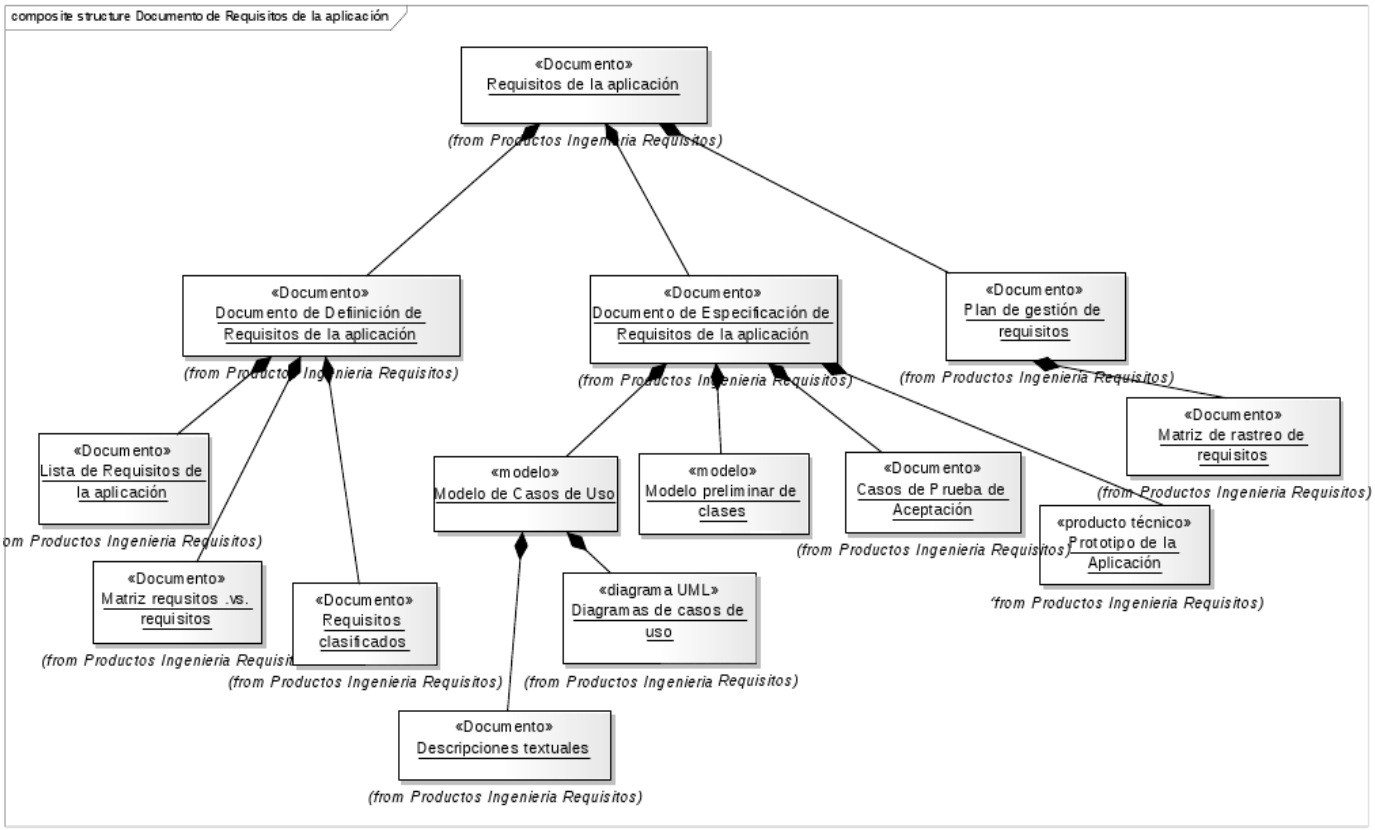

Figura 29. Estructura Documento Requisitos de la Aplicación
Fuente: Montilva y otros (2008)

Para el método VAREME se toma en cuenta sólo el Documento de Especificación de Requisitos, tal como lo indica Montilva y otros (2008), el cual es un documento técnico, que es el producto de entrada para el subproceso de Validación de Requisitos. El Documento de Especificación de Requisitos DER, se elabora en las etapas previas a la Validación de Requisitos, en los subprocesos de Descubrimiento, Analisis y Especificación de Requisitos, y está conformado por los requisitos modelados un conjunto de diagramas en UML, donde la primera actividad en el proceso de Validación de Requisitos es la revisión de los modelos que conforman dicho Documento. De este modo, Montilva y otros (2008) sugieren como modelos básicos el modelo de casos de uso y el modelo de clases para elaborar el prototipo, sin embargo, los otros métodos candidatos incorporan otros modelos en UML como son diagramas de actividades, de objetos, entre otros, como se especificó en la descripción de cada uno de ellos.

Es por ello que tomando las actividades de los tres métodos candidatos, esta fase se describe, inicialmente, de la siguiente manera, el producto de entrada para esta fase será el Documento de Especificación de Requisitos, ya que tomando como base el método de Montilva y otros (2008), este documento se comenzará a realizar en las fases anteriores al proceso de Validación de Requisitos, y se le harán los ajustes necesarios. Con lo cual el documento de especificación de requisitos pare el método VAREME está conformado por:

- Modelos de caso de uso (escenarios)
- Diagramas de actividades
- Diagrama de clases
- Modelo de navegación

Así, las actividades que conforman esta fase denominada Revisión Documento de Especificación de Requisitos son las siguientes:

- Validar la estructura del documento de especificación de requisitos
- Revisar especificación técnica de los modelos que conforman el documento de especificación de requisitos

ii. **Generación Prototipo de Interfaces de Usuario de Aplicaciones Web:** una vez realizada la revisión y validación de los modelos que conforman el Documento de Especificación de Requisitos, estos modelos son tomados como base para generar de forma semi-automática el prototipo de interfaces de usuario para aplicaciones Web, utilizando un software que permita la generación del código fuente y de las páginas HTML que conformarán un prototipo funcional. Las actividades correspondientes para esta fase son las siguientes:

- Generar Código y Páginas Web
- Generar Modelo de Datos (Base de datos relacional o NoSQL)

El prototipo generado, será evaluado por los usuarios finales para una retroalimentación. Con la retroalimentación, se refinan los requisitos de la aplicación que se está desarrollando en un proceso iterativo e incremental. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del evaluador, es decir del usuario. Esto permite que al mismo tiempo el Ingeniero de Software entienda mejor lo que se debe hacer y el usuario puede ver resultados a corto plazo, generando con esto, más confianza y ayudando a garantizar el éxito en el proceso de validación de requisitos.

Un aspecto que es necesario tener muy claro es el propósito del prototipo generado, el cual es probar los aspectos relacionados con la interacción del usuario final con el sistema, para validar requisitos. El prototipo en este caso, es una aplicación con la funcionalidad mínima para que el usuario pueda realizar las interacciones necesarias que permitan visionar el funcionamiento todavía ficticio del sistema resultante.

iii. **Validación de Requisitos a través del Prototipo de Interfaces de Usuario de Aplicaciones Web:** una vez generado el prototipo funcional, se procede a realizar las sesiones de validación de los requisitos con los usuarios finales, donde las actividades que componen esta fase según el análisis realizado en la matriz previa, son:

- Establecer sesiones de validación según los escenarios y roles
- Realizar sesiones de validación

En este paso los usuarios juegan un papel importante, su primer interés debe ser interactuar con el prototipo mediante experimentación para validar los requisitos. El Ingeniero de Software debe trabajar sistemáticamente y en conjunto con los usuarios finales para obtener y evaluar sus reacciones ante el prototipo, para luego incorporar las sugerencias e innovaciones de los usuarios ameriten en las modificaciones subsecuentes, de esta forma según la retroalimentación proporcionada por el usuario final, los modelos de ingeniería de software deben ser adaptados para incorporar los cambios surgidos, se genera nuevamente el prototipo en base a estos cambios y se repite el proceso de validación. Esta secuencia puede repetirse sistemáticamente hasta que exista un común entendimiento del dominio del problema, en un proceso iterativo e incremental

iv.Ajuste Documento de Especificación de Requisitos: según la retroalimentación proporcionada durante la las sesiones de validación, los modelos que conforman el Documento de Especificación de Requisitos deben ser adaptados para incorporar los cambios surgidos en estas sesiones, luego que se realicen estos cambios de debe verificar la consistencia e integridad de las especificaciones en base a los cambios realizados. Así las actividades para esta fase son las siguientes:

- Modificar modelos y descripciones
- Verificar consistencia e integridad de la Especificación Técnica

v. **Definición de Parámetros de Aceptación de la Aplicación:** Una vez que se hayan validado los requisitos a través del prototipo de interfaces de usuario, se deben definir los parámetros de aceptación y un documento de aceptación de requisitos, donde se logre el acuerdo con los usuarios finales. Las actividades que conforman esta fase son las siguientes:

- Determinar parámetros de aceptación de la aplicación.
- Verificar los parámetros de aceptación

Hasta este punto se tiene definido el flujo de trabajo del método VAREME, cada una de sus fases con las actividades correspondientes, por cada fase. Es importante destacar que según el objetivo planteado en la presente investigación el prototipo de interfaces de usuario para aplicaciones Web generado debe estar basado en patrones RIA, para lograr aumentar la capacidad de uso de las interfaces y las posibilidades de interacción de los usuarios.

Según, Busch y Koch (2009), la necesidad de un soporte de Ingeniería para el desarrollo de RIA ha sido abordada en los últimos años por varios métodos, de los cuales para la presente investigación se usará el enfoque basado en patrones, recomendaciones de diseños basados en las mejores prácticas, tal como es el caso de la librería de patrones propuesta por Scott y Neil (2009), explicada en la sección Bases Teóricas. Estos patrones resumen como los desarrolladores han logrado intercambiar principios conflictivos en el diseño de las interfaces de usuario, las cuales aunado con su experiencia derivan un conjunto de patrones de interfaces de usuario para la interacción enriquecida en la Web. Las RIA están centradas en el usuario, con lo cual los patrones se basan especialmente en la entrega de usabilidad para lograr una experiencia única en el usuario.

Desde que la usabilidad de la interfaz de usuario es un requisito fundamental, las aplicaciones Web usan las tecnologías RIA para cumplirlo. Según Valverde y Pastor (2009), las tecnologías RIA han introducido dos principales características en el diseño de las interfaces de usuario:

a) Widgets de interfaz de usuario RIA: La diferencia principal entre la interfaz de usuario producida utilizando HTML y las tecnologías RIA es el conjunto de widgets de interfaz de usuario disponibles. Las tecnologías RIA proveen una gran cantidad de widgets de interfaz de usuario similares a los usados en los entornos de escritorio (menús de navegación, datagrids, deslizadores, etc.). A través de los

widgets es posible implementar los patrones RIA previamente mencionados, para proveer ese aspecto enriquecido que caracteriza este tipo de aplicaciones Web.

b) Interacción dirigida por Eventos: las tecnologías RIA proporcionan interfaces de usuario altamente interactiva en fin de mejorar la experiencia del usuario. Esta interacción es provocada por los acontecimientos que son realizadas por el usuario final a través de los widgets de interfaz de usuario (por ejemplo, un clic del ratón, colocar el foco en un widget). Las posibles respuestas o reacciones a estos eventos varian desde un cambio en la interfaz de usuario a la la solicitud de los datos de la lógica de negocios. Las tecnologías RIA pueden proporcionar una amplia gama de eventos de interfaz de usuario y los lenguajes de programación específicos para ello.

En este sentido, existen un gran número de tecnologías que han surgido en la actualidad para el desarrollo de aplicaciones Web enriquecidas (RIA). Por un lado, existen un conjunto de librerías que pueden anexarse a las aplicaciones web tradicionales a fin de enriquecer parte de su interfaz de usuario. Las más utilizadas, según Blanco (2011), son las confeccionadas en lenguaje JavaScript, y algunos ejemplos están dados por jQuery[16], DOJO[17] y MooTools[18], por otros lado están los framework y lenguajes de programación, que utilizan como plataforma de ejecución HTML Dinámico y AJAX, los cuales brindan un mayor nivel de abstracción en este campo, al encapsular detalles de bajo nivel relativos al comportamiento de la interfaces de usuario e interacción, a través un modelo de componentes (widgets) y eventos a tal fin, que incorporan los patrones RIA previamente mencionados.

De este modo con cualquiera de los framework existentes en el mercado que tengan como objetivo facilitar el desarrollo de aplicaciones Web enriquecidas,

[16] http://jquery.com/
[17] http://www.dojotoolkit.org/
[18] http://mootools.net/

automatizando la generación de gran parte de la lógica de interfaz de usuario de la aplicación, incorporando los patrones RIA, se puede construir el prototipo a utilizar en el método VAREME.

Así, para cumplir con el objetivo de la presente investigación se debe incorporar al método obtenido hasta ahora la definición de las métricas a utilizar por cada uno de los factores de calidad a ser aplicados en la evaluación de la calidad del prototipo de interfaces de usuarios basado en patrones RIA, el cual es el paso que se detalla a continuación.

2. Definición de las métricas a utilizar en la evaluación de la calidad del prototipo de interfaces de usuarios basado en patrones RIA.

Como se mencionó anteriormente una definición correcta de los requisitos mejora la calidad de la aplicación a desarrollar, incluyendo tanto a los requisitos funcionales como a los no funcionales. Sin embargo, según Aguilar y otros (2011), en la mayoría de las metodologías de desarrollo de software solo se realiza el análisis de requisitos funcionales, ignorando por completo a los requisitos no funcionales hasta la fase de implementación. Este problema afecta directamente al usuario final: ya que no obtiene un producto que cumpla por completo con sus expectativas y no puede satisfacer sus necesidades con la aplicación. Por consiguiente, los requisitos no-funcionales deben ser analizados desde las fases iniciales del proceso de desarrollo con el fin de mejorar la calidad de la aplicación a desarrollar. Es por ello que en el método VAREME, algunos requisitos no funcionales son tomados en cuenta.

Así para evaluar la calidad del prototipo de interfaces de usuario basado en patrones RIA generado, luego de que el usuario final interactúe con él, se aplicará un instrumento de evaluación de la calidad del prototipo, el cual incorporará algunos requisitos no funcionales que se indicaran a continuación.

a. Seleccionar las características de calidad del modelo ISO/IEC 25010: en las Bases Teóricas se mencionaron, el estándar ISO 9126-1 y el ISO/IEC 25010, los

cuales proporcionan un marco conceptual para un modelo de calidad y presentan un conjunto de características y sub-características medibles. Estos proveen una base para especificar un modelo de calidad en diferentes dominios (como son los sitios y aplicaciones Web) y debe considerarse en cualquier enfoque de calidad en la Ingeniería de Software. Esta investigación se apoya en el modelo de calidad propuesto por la norma ISO/IEC 25010, la cual plantea el esquema del nuevo modelo de calidad para la evaluación de un producto de software.

De este modo, en base a la revisión de la literatura, las teorías sobre calidad del software, modelos y estándares de calidad analizados, se identificaron los elementos de calidad de un producto de software que pueden ser percibidos por los usuarios finales, tomados a partir de uno de los trabajos de investigación que se presentó como antecedente. Este trabajo es el de Macías y Gómez (2010), denominado "Utilizando el Modelo de Calidad de McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas de Información por los Usuarios", como ya se explicó previamente, define en base a los modelos McCall e ISO-9126 los principales factores que se pueden utilizar para evaluar un producto de software, considerando principalmente la integración de los usuarios finales en dicha evaluación.

Por lo cual partiendo de estos factores de calidad identificados por Macías y Gómez, (ver tabla 6), se definen los factores de calidad a ser usados en la presente investigación, los cuales se adaptaron al modelo de calidad ISO/IEC 25010, en base a las nuevas características y sub-características que este modelo incorpora. En este sentido, se estima que los factores de calidad a utilizar en la evaluación de la calidad del prototipo de interfaces de usuario basado en patrones RIA usado para validar requisitos del método VAREME, debe estar integrado por:

Tabla 15. Características de Calidad del Modelo ISO/IEC 25010 usados para Evaluar la Calidad por parte del Usuario Final, del Prototipo de Interfaces de Usuario basado en Patrones RIA del Método VAREME

Características de Calidad	Sub-característica	Descripción
Usabilidad	Facilidad de Entendimiento	El software es fácil de entender en su conveniencia y en cómo puede ser utilizado.
	Facilidad de Aprendizaje	El software permite al usuario aprender su uso de manera fácil y rápida, con eficacia, eficiencia y satisfacción en un determinado contexto de uso.
	Operabilidad	El software facilita al usuario su operación y control.
	Protección contra Errores del Usuario	El software protege a los usuarios de cometer errores.
	Estética de Interfaz de Usuario	El software posee interfaces de usuario agradables y permite satisfacer su interacción.
Funcionalidad	Corrección Funcional	El software proporciona resultados correctos y precisos.
	Adecuación Funcional	El software proporciona funciones que facilitan la realización de tareas específicas y objetivos.
Confiabilidad	Tolerancia a Fallas	El software está preparado para mantener un desempeño adecuado en caso de fallas.
	Recuperabilidad	El software que en caso de una interrupción o falla, pueda recuperar los datos directamente afectados y volver a establecer el estado deseado del sistema.
Eficiencia	Comportamiento en el Tiempo	El software cuenta con tiempos adecuados de respuestas al realizar sus diferentes funciones.

Fuente: La Autora (2012)

Una vez identificados las características, así como las sub-características o factores a evaluar, se procede a construir un instrumento que contenga los parámetros de medición para cada una de estos elementos. Para ello se utilizará un **cuestionario** el cual permitirá obtener información cualitativa sobre la opinión y la experiencia del usuario final. El cuestionario a utilizar tendrá un conjunto de preguntas cerradas, donde el usuario podrá seleccionar un conjunto de alternativas fijas, para ello se

requiere establecer una escala de valoración que permitan resultados de precisión, para permitir conclusiones útiles.

Para este caso la escala de valoración a utilizar es la escala de valoración numérica discreta, conocida también como escala de Likert, (Kirakowski, 2004), basada en el uso de números que representan divisiones de escala (intervalos de igual magnitud de medida). Esta escala permite conocer no solo si se cumple con el aspecto establecido, sino también en qué medida cree el evaluador que lo cumple. Las respuestas obtenidas pueden ser analizadas rápidamente.

Así para el instrumento a construir se utilizará una escala de Likert, con una granularidad de 1 a 5, que representan las divisiones de la escala, asumiendo intervalos de igual magnitud de medida: 1, 2, 3, 4, 5. De esta manera el entrevistado, el cual para este caso es el usuario final, expresa su nivel de aceptación o de rechazo refiriéndose a una escala que suele contar con 5 valores numéricos, los cuales son:

1: **Incumplido**: el prototipo no realiza la actividad o no muestra el contenido que ofrece.

2: **Pobremente cumplido:** lo evaluado desarrolla la actividad o muestra un contenido de una manera deficiente en incompleto.

3: **Medianamente cumplido**: el prototipo desarrolla la actividad o muestra un contenido relativamente útil, pero podría ser mejor.

4: **Cumplido:** el prototipo desarrolla la actividad o muestra un contenido útil.

5: **Totalmente cumplido:** el prototipo desarrolla la actividad o muestra un contenido útil que cumple o excede la expectativa del usuario

b. Definición de las métricas por cada característica de calidad a utilizar para la evaluación de la calidad: a continuación se va analizar cada una de las características de calidad seleccionadas del modelo ISO/IEC 25010, para establecer las métricas, para este caso las preguntas que se van a incorporar al cuestionario de evaluación a construir para ser utilizado como instrumento de

evaluación de calidad del prototipo de interfaces de usuario basado en patrones RIA utilizado en el método VAREME.

Evaluación de la Usabilidad

Existe un gran variedad de propuestas de evaluación basadas en modelos de usabilidad que siguen los estándares mencionados en la sección Bases Teóricas, de las cuales es importante resaltar las aportaciones orientadas a aplicaciones genéricas realizadas por Nielsen (1993) y Dromey (1998), ya que los estándares descritos han usado como base muchos trabajos para afianzar el concepto de usabilidad.

Nielsen (1993), uno de los autores más referenciados en el tema de la usabilidad, propuso un modelo de usabilidad bastante detallado centrado en los conceptos de aceptabilidad social y aceptabilidad práctica. En él se define la usabilidad como una sub-característica del concepto "utilidad", que es a su vez, una sub-característica de la aceptabilidad práctica. Nielsen mantiene que la usabilidad puede ser descompuesta en los siguientes atributos: facilidad de aprendizaje, eficiencia de uso, facilidad para recordar, errores mínimos, y atracción subjetiva.

Así, Dromey (1998) plantea un modelo de calidad basado en comportamientos y usos. Donde un comportamiento es algo que exhibe el propio software en un determinado contexto (por ejemplo, la usabilidad), y los usos son las acciones de los usuarios con el software. Este modelo enumera atributos de calidad específicos y los clasifica asociándolos a determinadas características propias del software.

Aunque los modelos anteriores asentaron las bases de la usabilidad para productos de software genéricos y el desarrollo de los estándares actuales, en lo concerniente a los productos de software orientados al ámbito Web hay que tener presentes características específicas que condicionan los aspectos a tener en cuenta en dichas sub-características pertenecientes a la usabilidad (Fernandez,

2009). Por este motivo, los modelos de calidad propuestos en estos estándares deben ampliarse y/o descomponerse en atributos medibles que además tengan en cuenta las características específicas de la Web. Esto ha motivado la aparición de propuestas de evaluación de usabilidad Web (y calidad, en general). Algunos de estos ejemplos los podemos encontrar en trabajos como los de Olsina y Rossi (2002), Granollers (2004), Ivory y Megraw (2005), Calero y otros (2005), Abrahão e Insfran (2006), Moraga y otros (2007) y Fernandez (2009).

En este orden de ideas, Nielsen uno de los autores que más han avanzado en la creación de criterios de medición de usabilidad, según Fernandez (2009), tempranamente tras la aparición del web, desarrolló los estudios necesarios para llegar a describir la existencia de un conjunto de principios de medición, cuyo cumplimiento permitirá asegurar la calidad de usabilidad de un Sitio Web. A dichos criterios les llamó principios heurísticos debido a que permitan hacer una evaluación considerando la perspectiva de los expertos. Estos principios están basados en 249 problemas de usabilidad, y han ido evolucionando desde sus trabajos realizados en Nielsen y Molich (1990) y Nielsen y Mack (1994). Estos principios se describen a continuación, Nielsen(2005):

- Visibilidad del estado del sistema: el sistema siempre deberá mantener informados a los usuarios de lo que está ocurriendo, a través de retroalimentación apropiada dentro de un tiempo razonable.
- Relación entre el sistema y el mundo real: el sistema deberá hablar el lenguaje de los usuarios mediante palabras, frases y conceptos que sean familiares al usuario, más que con términos relacionados con el sistema.
- Control y libertad del usuario: hay ocasiones en que los usuarios elegirán las funciones del sistema por error y necesitarán una salida de emergencia claramente marcada para dejar el estado no deseado al que accedieron, sin tener que pasar por una serie de pasos.

- Consistencia y estándares: los usuarios no deberán cuestionarse si acciones, situaciones o palabras diferentes significan en realidad la misma cosa; siga las convenciones establecidas.
- Prevención de errores: mucho mejor que un buen diseño de mensajes de error es realizar un diseño cuidadoso que prevenga la ocurrencia de problemas.
- Reconocimiento antes que recuerdo: las instrucciones para el uso del sistema deben estar a la vista o ser fácilmente recuperables cuando sea necesario.
- Flexibilidad y eficiencia de uso: la presencia de aceleradores, que no son vistos por los usuarios novatos, puede ofrecer una interacción más rápida a los usuarios expertos que la que el sistema puede proveer a los usuarios de todo tipo. Se debe permitir que los usuarios adapté el sistema para usos frecuentes.
- Estética y diseño minimalista: los diálogos no deben contener información que es irrelevante o poco usada. Cada unidad extra de información en un diálogo, compite con las unidades de información relevante y disminuye su visibilidad relativa.
- Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: los mensajes de error se deben entregar en un lenguaje claro y simple.
- Ayuda y documentación: incluso en los casos en que el sistema pueda ser usado sin documentación, podrá ser necesario ofrecer ayuda y documentación.

Por lo anteriormente expuesto para el diseño del instrumento de evaluación de la característica usabilidad del prototipo de interfaces de usuario para el método VAREME, se basa en el trabajo correspondiente a la evaluación de usabilidad Web de Fernandez (2009), el cual parte del modelo de Abrahão e Insfran (2006) para productos de software genéricos, que se extendió y adapto a productos orientados a la Web basado en los principios heurísticos de Nielsen (2005) y a la norma ISO/IEC 25010. Para ello el modelo de usabilidad Web, tuvo en cuenta la división de sub-características que se realiza en la ISO/IEC 25010 para la característica usabilidad, subdividiéndolas en atributos cercanos al dominio Web, con el objetivo

de obtener un modelo de usabilidad lo más amplio posible centrado en el ámbito de aplicaciones Web.

De esta manera en base a las cinco (5) sub-características seleccionadas para la característica usabilidad, del modelo de calidad ISO/IEC 25010 (especificadas en la tabla 15), y a los atributos definidos para cada sub-característica según Fernandez (2009), las preguntas que formarán parte del cuestionario de evaluación de calidad para el prototipo de interfaces de usuario del método VAREME, se detallan en las siguientes tablas:

Tabla 16. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica "Facilidad de Entendimiento".

Atributo	Factor	Pregunta
Legibilidad Visual	Adecuación de la fuente y Disposición	¿El texto es legible? ¿El tamaño de la fuente es adecuado al contexto?
	Adecuación de la visualización textual	¿La combinación de colores de texto y su fondo son adecuados?
Facilidad de Lectura	Densidad de Información	¿El espacio físico usado es el adecuado? ¿La cantidad de información está bien distribuida?
Familiaridad	Consistencia de formato	¿Uso de formatos distintos para datos? (Ejemplo: Fecha dd/mm/aaaa) ?
	Internacionalización	¿Se hace uso de elementos y formas que siguen estándares?
	Comodidad	¿La aplicación facilita que el usuario se sienta cómodo?
Ahorro de Esfuerzo	Acciones mínimas	¿Se dispone de valores por defecto? ¿Las acciones se pueden realizar sencillamente en pocos pasos?
	Auto-descripción	¿Existe completitud en las descripciones? ¿Existe claridad de los elementos que conforman la interfaz?
Orientación al Usuario	Información clara	¿El título de una página describe con precisión su objetivo? ¿En el prototipo se usa un lenguaje sencillo?
	Calidad de los mensajes de actualización	¿Los mensajes de actualización son significativos para el usuario?

	Calidad de los mensajes de aviso	¿Los mensajes avisan adecuadamente la acción que el usuario va a llevar a cabo?
	Orientación en las imágenes	¿Existe texto alternativo en las imágenes que se muestran?
	Mensajes de Orientación	¿Se indican que campos son obligatorios en un formulario? ¿En los campos con formatos se indica el formato a usar?
Navegabilidad	Soporte de búsqueda interna	¿El prototipo Web permite alcanzar contenidos si navegar explícitamente por los enlaces determinados a alcanzar ese contenido?
	Clickabilidad	¿Los enlaces que generan una acción o contenido son claramente reconocidos?
	Alcanzabilidad	¿Es fácil acceder a los contenidos y/o acciones del prototipo Web? ¿El menú es consistente en toda la aplicación? ¿Todos los vínculos funcionan?

Fuente: La Autora (2012)

Tabla 17. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica "Facilidad de Aprendizaje".

Atributo	Factor	Pregunta
Predictibilidad	Nombres de enlaces significativos	¿Los nombres de los enlaces son adecuados y significativos?
	Controles significativos	¿Los controles están seleccionados adecuadamente para cada función?
Potencialidad	Determinación de acciones posibles	¿Es fácil reconocer de forma rápida y clara qué acciones puede realizar el usuario en una interfaz?
Retroalimentación Informativa	Progreso explicito de las transacciones	¿Existen elementos que muestran el progreso de una transacción?
	Contexto explicito del usuario	¿Existen elementos que permitan al usuario saber exactamente dónde se encuentra dentro de la aplicación y cómo volver atrás (breadcrumbs)?
	Controles	¿Los controles de interfaces, proporcionan ayuda o información de su uso?

Fuente: La Autora (2012)

Tabla 18. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica "Operabilidad".

Atributo	Factor	Pregunta
Compatibilidad	Compatibilidad con los navegadores	¿El prototipo Web puede ser ejecutado en los navegadores más comunes sin alterar su comportamiento y apariencia?
	Compatibilidad con los sistemas operativos	¿El prototipo Web puede ser visualizado en los sistemas operativos más comunes sin alterar su comportamiento y apariencia?
	Compatibilidad con la resolución de pantalla	¿El prototipo Web se adapta a las resoluciones de pantalla más comunes?
Gestión de Datos	Operabilidad de los datos	¿Los datos se visualizan de forma completa y sencilla? ¿Se pueden realizar acciones sobre los datos fácilmente? ¿Se pueden realizar búsquedas para acceder a los datos rápidamente?
Controlabilidad	Edición posterior	¿El contenido introducido por el usuario es editable? ¿Se permite la corrección de errores en los datos de entrada?
	Soporte a operaciones de cancelación	¿Las acciones se pueden cancelar sin efectos perjudiciales al funcionamiento normal?
Capacidad de Adaptación	Adaptativo	¿El prototipo Web puede adaptarse a las necesidades de los distintos usuarios?
Consistencia	Diseño Consistente	¿El diseño es consistente en todas las pantallas del prototipo?
	Comportamiento constante de los enlaces	¿Los enlaces con el mismo nombre siempre apuntan al mismo destino?
	Comportamiento constante de los controles	¿Los controles del mismo tipo mantienen el mismo comportamiento?
	Permanencia de los enlaces	¿Los enlaces se mantienen siempre en la misma posición de la interfaz?
	Consistencia en el orden de los enlaces	¿Los enlaces pertenecientes a un mismo grupo siempre aparecen en el mismo orden?

	Consistencia en las etiquetas	¿Las etiquetas se corresponden con el campo que hacen referencia?

Fuente: La Autora (2012)

Tabla 19. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica "Protección contra errores del usuario".

Atributo	Factor	Pregunta
Gestión de Errores	Prevención de errores	¿Se proveen mecanismos de validación de datos de entrada? ¿Se proveen mecanismos que faciliten al usuario la entrada de datos?
	Calidad de los mensajes de error	¿Los mensajes de error representan de forma clara y concisa el error ocurrido? ¿Los mensajes de error sugieren una solución al problema ocurrido?
	Calidad de los mensajes de ayuda	¿Los mensajes de ayuda son claros y concisos?

Fuente: La Autora (2012)

Tabla 20. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a la sub-característica "Estética de la Interfaz de Usuario".

Atributo	Factor	Pregunta
Grado de Atracción	Uniformidad de color de fondo	¿Los colores de fondo empleados en los elementos de las interfaces el usuario son siempre los mismos? ¿Pueden elementos de primer plano (ya sea texto o imágenes) distinguirse fácilmente del fondo?
	Uniformidad de la fuente	¿El color, estilo y tipo de fuente empleados en los elementos de las interfaces se mantienen uniformes?
	Uniformidad en la posición de las secciones de la interfaz	¿Existen elementos no alineados o desordenados? ¿Las secciones en las que se divide la interfaz, se mantienen uniformes en toda la aplicación (prototipo)?

Fuente: La Autora (2012)

Evaluación de la Funcionalidad, Confiabilidad y Eficiencia

Partiendo del estándar ISO/IEC 25010 y del trabajo de Macías y Gómez (2010), las preguntas que forman parte del cuestionario de evaluación de calidad para el prototipo de interfaces de usuario del método VAREME, correspondientes a las características funcionalidad, confiabilidad y eficiencia se detallan en la siguiente tabla:

Tabla 21. Preguntas del Cuestionario de Evaluación de Calidad, correspondiente a las característica Funcionalidad, Confiabilidad y Eficiencia

Característica	Sub-característica	Pregunta
Funcionalidad	Corrección Funcional	¿El prototipo realiza las funciones acordadas en la forma esperada y correcta? ¿El prototipo alcanza los objetivos especificados?
	Adecuación Funcional	¿El prototipo tiene el conjunto de funciones apropiadas para las tareas especificadas?
Confiabilidad	Tolerancia a Fallas	¿El usuario puede interrumpir una operación sin afectar el funcionamiento normal del prototipo Web? ¿En caso de ocurrir un error, el prototipo Web sigue funcionando de forma normal?
	Recuperabilidad	¿El prototipo Web es capaz de volver a un estado estable tras un error ocurrido?
Eficiencia	Comportamiento en el Tiempo	¿Los usuarios realizan sus tareas correctamente en el menor tiempo posible? ¿Las acciones que y tareas están diseñada para realizarse de la forma más rápida e intuitiva posible?

Fuente: La Autora (2012)

Una vez definidas las preguntas que formarán parte del cuestionario, se procede a unificar todas las preguntas con la escala de valoración definida anteriormente, de esta forma el cuestionario resultante se presenta a continuación:

Tabla 22. Cuestionario de Evaluación de Calidad del Prototipo de Interfaces de Usuario basado en Patrones RIA del Método VAREME

Preguntas		Valoración				
		1 Incumplido	2 Pobremente Cumplido	3 Medianamente Cumplido	4 Cumplido	5 Totalmente Cumplido
Usabilidad						
Facilidad de Entendimiento						
1	¿El texto es de las interfaces de usuario es legible?					
2	¿El tamaño de la fuente es adecuado al contexto?					
3	¿La combinación de colores de texto y su fondo son adecuados?					
4	¿El espacio físico usado es el adecuado?					
5	¿La cantidad de información está bien distribuida?					
6	¿Uso de formatos distintos para datos? (Ejemplo: Fecha dd/mm/aaaa) ?					
7	¿Se hace uso de elementos y formas que siguen estándares?					
8	¿Se dispone de valores por defecto?					
9	¿La aplicación facilita que el usuario se sienta cómodo?					
10	¿Las acciones se pueden realizar sencillamente en pocos pasos?					
11	¿Existe completitud en las descripciones?					
12	¿Existe claridad de los elementos que conforman la interfaz?					
13	¿El título de una página describe con precisión su objetivo?					
14	¿En el prototipo se usa un lenguaje sencillo?					
15	¿Los mensajes de actualización son significativos para el usuario?					
16	¿Los mensajes avisan adecuadamente la acción que el usuario va a llevar a cabo?					

Preguntas		Valoración				
		1 Incumplido	2 Pobremente Cumplido	3 Medianamente Cumplido	4 Cumplido	5 Totalmente Cumplido
17	¿Existe texto alternativo en las imágenes que se muestran?					
18	¿Se indican que campos son obligatorios en un formulario?					
19	¿En los campos con formatos se indica el formato a usar?					
20	¿El prototipo Web permite alcanzar contenidos sin navegar explícitamente por los enlaces determinados a alcanzar ese contenido?					
21	¿Los enlaces que generan una acción o contenido son claramente reconocidos					
22	¿Es fácil acceder a los contenidos y/o acciones del prototipo Web?					
23	¿El menú es consistente en toda la aplicación?					
24	¿Todos los vínculos funcionan?					
Facilidad de Aprendizaje						
25	¿Los nombres de los enlaces son adecuados y significativos?					
26	¿Los controles están seleccionados adecuadamente para cada función?					
27	¿Es fácil reconocer de forma rápida y clara qué acciones puede realizar el usuario en una interfaz?					
28	¿Existen elementos que muestran el progreso de una transacción?					
29	¿Existen elementos que permitan al usuario saber exactamente dónde se encuentra dentro de la aplicación?					
30	¿Los controles de interfaces, proporcionan ayuda o información de su uso?					
Operabilidad						
31	¿El prototipo puede ser ejecutado en los navegadores más comunes sin alterar su comportamiento y apariencia?					

Preguntas		Valoración				
		1 Incumplido	2 Pobremente Cumplido	3 Medianamente Cumplido	4 Cumplido	5 Totalmente Cumplido
32	¿Los controles del mismo tipo mantienen el mismo comportamiento?					
33	¿El prototipo Web puede ser visualizado en los sistemas operativos más comunes sin alterar su comportamiento y apariencia?					
34	¿El prototipo Web se adapta a las resoluciones de pantalla más comunes?					
35	¿Los datos se visualizan de forma completa y sencilla?					
36	¿Se pueden realizar acciones sobre los datos fácilmente?					
37	¿Se pueden realizar búsquedas para acceder a los datos rápidamente?					
38	¿El contenido introducido por el usuario es editable?					
39	¿El prototipo Web puede adaptarse a las necesidades de los distintos usuarios?					
40	¿El diseño es consistente en todas las pantallas del prototipo?					
41	¿Los enlaces con el mismo nombre siempre apuntan al mismo destino?					
42	¿Los enlaces se mantienen siempre en la misma posición de la interfaz?					
43	¿Los enlaces pertenecientes a un mismo grupo siempre aparecen en el mismo orden?					
44	¿Las etiquetas se corresponden con el campo que hacen referencia?					

Preguntas		Valoración				
		1 Incumplido	2 Pobremente Cumplido	3 Medianamente Cumplido	4 Cumplido	5 Totalmente Cumplido
Protección contra Errores del Usuario						
45	¿Se proveen mecanismos de validación de datos de entrada?					
46	¿Se proveen mecanismos que faciliten al usuario la entrada de datos?					
47	¿Los mensajes de error representan de forma clara y concisa el error ocurrido?					
48	¿Los mensajes de error sugieren una solución al problema ocurrido?					
49	¿Los mensajes de ayuda son claros y concisos?					
Estética de la Interfaz de Usuario						
50	¿Los colores de fondo empleados en los elementos de las interfaces son siempre los mismos?					
51	¿Pueden elementos de primer plano (ya sea texto o imágenes) distinguirse fácilmente del fondo?					
52	¿El color, estilo y tipo de fuente empleados en los elementos de las interfaces se mantienen uniformes?					
53	¿Existen elementos no alineados o desordenados?					
54	¿Las secciones en las que se divide la interfaz, se mantienen uniformes en toda la aplicación (prototipo)?					
Funcionalidad						
Corrección Funcional						
55	¿El prototipo realiza las funciones acordadas en la forma esperada y correcta?					
56	¿El prototipo alcanza los objetivos especificados?					

Preguntas		Valoración				
		1 Incumplido	2 Pobremente Cumplido	3 Medianamente Cumplido	4 Cumplido	5 Totalmente Cumplido
Adecuación Funcional						
57	¿El prototipo tiene el conjunto de funciones apropiadas para las tareas especificadas?					
Fiabilidad						
Tolerancia a Fallas						
58	¿El usuario puede interrumpir una operación sin afectar el funcionamiento normal del prototipo Web?					
59	¿En caso de ocurrir un error, el prototipo Web sigue funcionando de forma normal?					
Recuperabilidad						
60	¿El prototipo Web es capaz de volver a un estado estable tras un error ocurrido?					
Eficiencia						
Comportamiento en el Tiempo						
61	¿Los usuarios realizan sus tareas correctamente en el menor tiempo posible?					
62	¿Las acciones que y tareas están diseñada para realizarse de la forma más rápida e intuitiva posible?					

Fuente: La Autora (2012)

Una vez definido el cuestionario que se utilizará para validar el prototipo de interfaces de usuario basado en patrones RIA del método VAREME, es necesario especificar la medición de los resultados obtenidos una vez que el instrumento sea aplicado a los usuarios finales involucrados en el proceso de validación de requisitos. Como la escala de medición usada fue la escala de likert, el procedimiento para obtener los resultados, según Kirakowski, (2004), es sumando los valores obtenidos respecto de cada pregunta y compararlo con el puntaje máximo y mínimo. El puntaje mínimo resulta de la multiplicación del número de ítems por 1 y el puntaje máximo

está dado por el número de ítems o preguntas multiplicado por 5. De este modo una puntuación se considera alta o baja respecto a qué valor se acerque más el resultado obtenido.

c. **Asociar los patrones RIA a los parámetros de calidad definidos para la característica usabilidad**: para los factores de calidad definidos para la característica usabilidad, se pueden asociar patrones RIA que sean incorporados a las interfaces de usuario del prototipo, para ayudar a garantizar el cumplimento de dichos factores de calidad y así cumplir con la usabilidad del mismo. El desarrollador puede usar estas recomendaciones que se indican a continuación e incorporar estos patrones al momento de generar el prototipo de interfaces de usuario, como una guía que le ayude a cumplir con la característica de calidad usabilidad. Los patrones RIA que se especifican a continuación están tomados del trabajo de Scott y Neil (2009).

Tabla 23. Patrones RIA recomendados usar, asociados a los factores de calidad identificados para medir la característica Usabilidad del modelo ISO/IEC 25010

Atributo	Factor	Patrón RIA Recomendado
Facilidad de Entendimiento		
Familiaridad	Consistencia de formato	Principio Ser Directo Patrón Selección Directa • Uso del control calendario para las Fechas. • Uso de control con mascaras que den formato
Ahorro de Esfuerzo	Acciones mínimas	Principio Mantenerlo ligero Patrón Herramientas Contextuales • Herramientas siempre visibles • Herramientas que se revelan con el pase del ratón (mouse hover) • Herramientas que se revelan u ocultan con una acción • Herramientas multinivel • Menú secundario • Proveer atajos de teclado (combinación de teclas) para realizar una acción

		Principio Ser Directo Patrón Editar en la página Patrón Arratrar y Soltar (Drag and drop) Patrón Selección Directa • Selección alterna • Selección usando checkbox • Selección que abarca varias páginas. • Selección de objetos
	Auto-descripción	Principio Proporcionar una Invitación Patrón Invitaciones Estáticas y Dinámicas • Invitación con ToolTipText • Invitación al pasar el ratón sobre el control o etiqueta Principio Permanecer en la Página • Patrón Overlay: superponer los mensajes de ayuda o ilustraciones directamente sobre la interfaz
Navegabilidad	Soporte de búsqueda interna Clickabilidad Alcanzabilidad	Principio Permanecer en la Página Patrón Overlay, Inlay, Flujo de Procesos y Patrón Páginas Virtuales: • Scrolling Virtual • Paginación en Linea • Carrusel • Pestañas Principio Mantenerlo ligero Patrón Herramientas Contextuales • Menú secundario • Menú de navegación • Breadcrumbs (migas de pan)
Facilidad de Aprendizaje		
Potencialidad	Determinación de acciones posibles	Principio Proporcionar una Invitación Patrón Invitaciones Estáticas y Dinámicas • Invitación con ToolTipText • Invitación al pasar el ratón sobre el control o etiqueta
Retroalimentación Informativa	Progreso explicito de las transacciones	Principio Reaccionar Inmediatamente Patrones de Retroalimentación • Vista previa activa • Indicador de progreso • Revelación progresiva • Actualización Periódica • Mensajes de retroalimentación

Operabilidad		
Gestión de Datos	Operabilidad de los datos	Principio Mantenerlo ligero Patrón Herramientas Contextuales • Tabla (Grid) para presentar los datos o Ordenar por columnas o Selección e interacción con elementos o Paginación o Filtros
Controlabilidad	Edición posterior	Principio Ser Directo Patrón Editando en la página • Edición en línea de un solo campo • Edición en línea de múltiples campos • Editando en un panel superpuesto • Editando directamente en la tabla (grid)
Prevención Contra Errores de Usuario		
Gestión de Errores	Prevención de errores en datos de entrada	Principio Reaccionar Inmediatamente Patrones de Búsqueda • Autocompletar • Sugerencias de búsqueda • Búsqueda refinada

Fuente: La Autora (2012)

De este modo, a continuación se propone un instrumento para evaluar el uso de Patrones RIA en el prototipo de interfaces de usuario generado para el método VAREME, en base a los patrones RIA recomendados a usar (indicados en la tabla anterior) y según los patrones más usados en aplicaciones Web según Scott y Neil (2009). Se propone este instrumento de evaluación, el cual será aplicado al ingeniero de software, para así garantizar que el prototipo a utilizar en el método VAREME cumpla al menos con mínimo de patrones para poder considerarse un prototipo basado en RIA.

Este instrumento será un cuestionario con un conjuntos de preguntas cerradas, cuyas respuestas serán SI o NO, para identificar si se usan los patrones RIA recomendados. El cuestionario se especifica a continuación:

Tabla 24. Cuestionario de Evaluación del Uso de Patrones RIA recomendados en el Prototipo de Interfaces de Usuario a Utilizar en el Método VAREME

Preguntas		Valoración	
		SI	NO
1	¿Uso del control calendario para las Fechas?		
2	¿Uso de control con mascaras que den formato?		
3	¿Uso de herramientas que se revelan con el pase del ratón (mouse hover)?		
4	¿Uso de herramientas que se revelan u ocultan con una acción?		
5	¿Uso de Menú secundario?		
6	¿Proveer atajos de teclado (combinación de teclas) para realizar una acción?		
7	¿Se permite la selección alterna de elementos?		
8	¿Se permite la selección de elementos usando checkbox?		
9	¿Se puede arratrar y soltar elementos (Drag and drop)?		
10	¿Uso de ToolTipText?		
11	¿Uso de Overlay: superponer los mensajes de ayuda o ilustraciones directamente sobre la interfaz?		
12	¿Uso de menús de navegación?		
13	¿Uso de menú Breadcrumbs (migas de pan)?		
14	¿Vista previa activa?		
15	¿Uso de indicador de progreso?		
16	¿Actualización periódica de los datos?		
17	¿Mensaje retroalimentación cada vez que se ejecuta una acción?		
18	¿Uso de tabla (Grid) para presentar los datos?		
19	¿ Selección e interacción con datos de la tabla (grid)?		
20	¿Uso de paginación en la tabla (grid) de datos?		
21	¿Uso de filtros o búsquedas en la tabla (grid) de datos?		
22	¿Editando registros directamente en la tabla (grid)?		
23	¿Uso del control autocompletar?		
24	¿Búsquedas avanzadas?		

Fuente: La Autora (2012)

Con el uso de este instrumento, se puede comprobar si el prototipo generado a usar en el método de VAREME, cumple con los requisitos mínimos necesarios para que pueda ser considerado un prototipo basado en patrones RIA y por lo tanto ayudar a garantizar la usabilidad por parte del usuario.

Al usar este cuestionario y evaluar los resultados, se considera que el prototipo de interface de usuarios se considera un prototipo basado en patrones RIA si cumple al menos con el 70 % de respuestas afirmativas.

3. Método VAREME.

Finalmente una vez determinado el flujo de trabajo a seguir en el método y definidas las métricas a usar por cada uno de los factores de calidad a ser aplicados y los patrones RIA asociados a los mismos, se procede a la integración de estos elementos para el diseño final del método VAREME.

Es importante destacar que a algunas de las fases del método VAREME definidas anteriormente, se les incorporó nuevas actividades relacionadas con los patrones RIA y el enfoque de calidad, estas fases son:

- La fase 2 denominada Generación de Prototipo de Interfaces de Usuario de Aplicaciones Web, se modifico el nombre por: "Generación de Prototipo de Interfaces de Usuario basado en Patrones RIA". A esta fase también se incorporo la actividad siguiente:
  - Evaluar los patrones RIA del prototipo de interfaces de usuario por parte del ingeniero de software.
- La fase 5 denominada Definición de Parámetros de Aceptación de la Aplicación, a la cual se añade la actividad:
  - Evaluar la calidad del prototipo a través de un cuestionario por parte de los usuarios finales

El flujo de trabajo completo del método VAREME se detalla a continuación a través de un diagrama de productos y procesos (PDD), así como también de una tabla donde se especifica y se describen cada una de las actividades y sub-actividades del método VAREME, finalmente se presentan los productos resultantes en cada actividad.

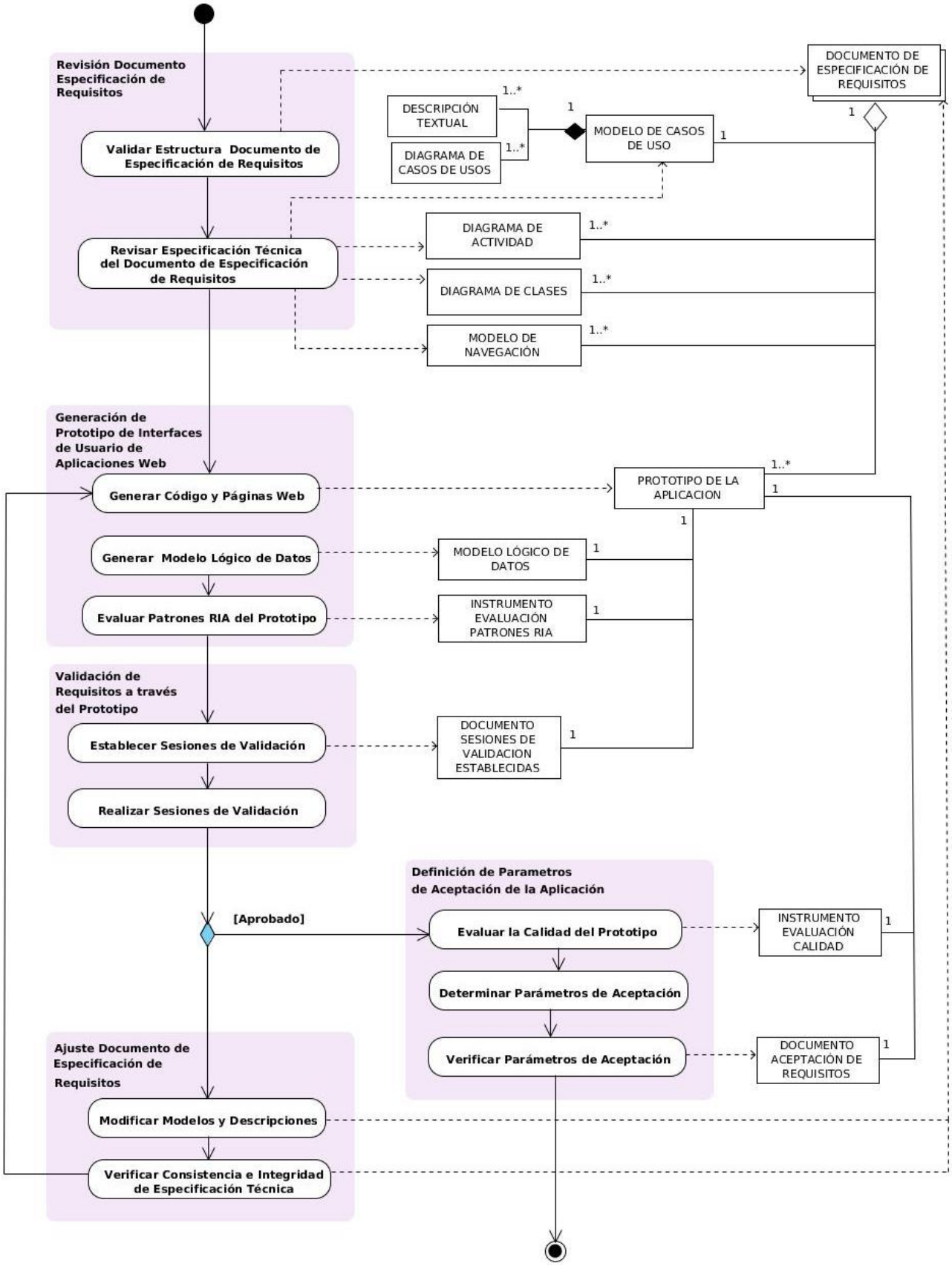

Figura 30. PDD del Método VAREME
Fuente: La Autora (2012)

Tabla 25. Actividades y Sub-Actividades del Método VAREME

Actividad	Sub-Actividad	Descripción
Revisión Documento de Especificación de Requisitos	Validar la estructura del documento de especificación de requisitos	Revisión y validación de la estructura y contenido del DOCUMENTO DE ESPECIFICACION DE REQUISITOS, al cual se realizan las correcciones pertinentes si lo amerita.
	Revisar especificación técnica de los modelos que conforman el documento de especificación de requisitos	En esta actividad se realiza la revisión de la especificación técnica de los modelos que conforman El DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS: • MODELO DE CASOS DE USO • DIAGRAMA DE ACTIVIDADES • DIAGRAMA DE CLASES • MODELO DE NAVEGACIÓN Cada uno de estos modelos servirá como base para la generación del PROTOTIPO DE LA APLICACIÓN, en este paso se revisa cada uno de ellos y se realizan las correcciones pertinentes si así lo amerita.
Generación Prototipo de Interfaces de Usuario de Aplicaciones Web	Generar Código y Páginas Web	A partir de los modelos que conforman el DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS se genera el PROTOTIPO DE LA APLICACIÓN. Para generar el prototipo se puede usar cualquier framework o herramienta existente en el mercado que tenga como objetivo facilitar el desarrollo de aplicaciones Web enriquecidas, y que facilite la automatización de la generación de gran parte de la lógica de interfaz de usuario de la aplicación, incorporando los patrones RIA. Al momento de generar el PROTOTIPO el Ingeniero de Software se debe apoyar en el documento de Patrones RIA recomendados a usar, (ver tabla 23) propuesto en la presente investigación El resultado de este proceso es un conjunto de páginas HTML que conforman un PROTOTIPO funcional basado en patrones RIA.

	Generar Modelo Lógico de Datos	Generación de un modelo lógico de datos, ya sea una base de datos relacional o NoSQL, el cual ayuda a proveer la funcionalidad del PROTOTIPO DE LA APLICACIÓN. Las clases definidas en el DIAGRAMA DE CLASES pueden ayudar a generar el modelo de datos en forma automática, en el caso de base de datos relacionales las clases se convierten en tablas que representan la estructura de datos, y sus atributos los campos respectivos. Para el caso de un modelo NoSQL las clases serán documentos o colecciones de datos.
	Evaluar los patrones RIA del prototipo de interfaces de usuario, por parte del ingeniero de software.	Una vez generado el PROTOTIPO DE LA APLICACIÓN, se recomienda revisar los patrones RIA incorporados en el mismo. Pare ello el ingeniero de software evalúa el PROTOTIPO de interfaces de usuario generado a través de un cuestionario (ver tabla 24), con el fin de comprobar si el PROTOTIPO cumple con los requisitos mínimos necesarios para que pueda ser considerado un prototipo basado en patrones RIA y así poder continuar con el flujo de actividades del método VAREME.
Validación de Requisitos a través del Prototipo de Interfaces de Usuario basado en Patrones RIA	Establecer sesiones de validación según los escenarios y roles	En base a los escenarios y roles definidos en los modelos de caso de uso, se establecen y organizan las SESIONES DE VALIDACIÓN. Para cada sesión según el(los) escenario(s), se seleccionan uno o un conjunto de usuarios representativo de los distintos roles definidos, de tal modo que sea posible validar el PROTOTIPO DE LA APLICACIÓN en todos sus modos de utilización.
	Realizar sesiones de validación	Realizar las SESIONES DE VALIDACIÓN, para validar los requisitos a través de la interacción de los usuarios finales con el PROTOTIPO DE LA APLICACIÓN. Según la retroalimentación proporcionada durante la interacción con el PROTOTIPO DE LA APLICACIÓN, los modelos del DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS deben ser adaptados para incorporar los cambios surgidos, se genera nuevamente el PROTOTIPO DE LA APLICACIÓN en base a estos cambios, y se repite el proceso de validación.

		Esta secuencia puede repetirse sistemáticamente hasta que exista un común entendimiento del dominio del problema. Es importante tener claro que el PROTOTIPO es una aplicación con la funcionalidad mínima para que el usuario pueda realizar las interacciones necesarias que permitan visionar el funcionamiento todavía ficticio del sistema resultante, no es el sistema final.
Ajuste Documento de Especificación de Requisitos	Modificar modelos y descripciones	Según la retroalimentación proporcionada durante la las sesiones de validación, los modelo que forman parte del DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS deben ser adaptados para incorporar los cambios surgidos en estas sesiones.
	Verificar consistencia e integridad de la especificación técnica.	En base a los resultados de la validación de requisitos y los ajustes a los modelos correspondientes, se verifica la consistencia e integridad de la especificación técnica de los modelos que conforman el DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS. Una vez realizado los cambios correspondientes en el DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS, se genera nuevamente el PROTOTIPO DE LA APLICACIÓN en base a estos cambios, y se repite el proceso de validación.
Definición de Parámetros de Aceptación de la Aplicación	Evaluar la calidad del prototipo de la aplicación a través de un cuestionario, por parte de los usuarios finales	Una vez que el usuario haya interactuado con el prototipo en las sesiones de validación, se procederá a evaluar la calidad del PROTOTIPO por parte del usuario, a partir de un instrumento que incorpora algunos factores de calidad del modelo de calidad ISO/IEC 25010 (ver tabla 22). Esta evaluación se realiza con el fin de mejorar la calidad del producto final y satisfacer las necesidades del usuario.
	Determinar parámetros de aceptación de la aplicación.	Definir parámetros de aceptación a utilizar para lograr un acuerdo con los usuarios finales luego de haber validado los requisitos a través del PROTOTIPO DE LA APLICACIÓN.

	Verificar los parámetros de aceptación.	Verificar, con los interesados, los parámetros de aceptación de la aplicación y una vez llegado a un acuerdo generar el DOCUMENTO DE ACEPTACIÓN DE REQUISITOS y culminar el proceso de validación.

Fuente: La Autora (2012)

Tabla 26. Productos del Método VAREME

Concepto		Descripción
DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS	MODELO DE CASOS DE USO (ESCENARIO)	EL Documento de Especificación de Requisitos describe el conjunto de requisitos que establecen los usuarios de la aplicación a través de un conjunto de modelos elaborados usando la notación UML. Está dirigida al Grupo de Diseño que tiene a su cargo elaborar el diseño de la aplicación, por consiguiente, tiene un carácter técnico (Montilva y otros, 2008) El Modelo de Casos de Uso: Describen gráfica y textualmente la funcionalidad del sistema, el conjunto de funciones que el sistema ofrecerá al usuario a través de su interfaz. Los casos de uso especifican la manera en que los actores interactúan con el sistema, y describen los escenarios que serán percibidos de distinta forma por distintos actores. El modelo de casos de uso esta conformador por: la descripción textual y los diagramas de casos de uso (Pressman, 2010).
	DIAGRAMA DE ACTIVIDAD	Representa el flujo de interacción con respecto a un escenario específico (Pressman, 2010).

	DIAGRAMA DE CLASES	En un diagrama de estructura estática que muestra las clases del sistema y sus interrelaciones. Diagramas de clases son el pilar básico del modelado en UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño) (Pressman, 2010).
	MODELO DE NAVEGACIÓN	Su objetivo es definir cómo se le proporcionará a cada usuario del sistema el acceso a la información y la funcionalidad que le es relevante para llevar a cabo su tarea dentro del sistema y qué secuencias de caminos deberá seguir para conseguirlo. (Ogata y Matsuura, 2010).
PROTOTIPO DE LA APLICACIÓN		Es un una versión inicial de un sistema de software que se utiliza para demostrar conceptos, probar opciones de diseño y, en general, informarse más del problema y sus posibles soluciones (Sommerville , 2005)
MODELO LÓGICO DE DATOS		Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior (Codd, 1970).
INSTRUMENTO EVALUACIÓN PATRONES RIA		Es un cuestionario con un conjunto de preguntas que tiene por objetivo evaluar un prototipo de interfaces de usuarios para aplicaciones Web con el fin de comprobar que cumple con los requisitos mínimos necesarios para que pueda ser considerado un prototipo basado en patrones RIA .
DOCUMENTO SESIONES DE VALIDACIÓN ESTABLECIDAS		Es un documento donde se especifica las sesiones a realizar para validar requisitos a través de un prototipo. En base a los escenarios y roles definidos en los modelos de caso de uso, se establecen y organizan estas sesiones donde se indican un conjunto de usuarios representativo de los distintos roles definidos (Gabrysiak y otros, 2011).
INSTRUMENTO EVALUACION DE CALIDAD DEL PROTOTIPO		Es un cuestionario con un conjunto de preguntas que tiene por objetivo evaluar la calidad de un prototipo de interfaces de usuario basado en patrones RIA. Está basado en algunos factores de calidad del estándar ISO/IEC 25010.

DOCUMENTO ACEPTACIÓN DE REQUISITOS	Es un documento el cual, luego del proceso de validación de requisitos, permite dejar constancia del acuerdo logrado, entre el ingeniero de software y los interesados (cliente y usuarios finales) (Gabrysiak y otros, 2011).

Fuente: La Autora (2012)

4. Validación del método VAREME ejemplificando su uso a partir de un caso de estudio.

Con el objetivo de ejemplificar el método propuesto, se plantea el siguiente caso de estudio basado en un ejemplo sencillo, el cual se define a continuación:

Chacaticos es una empresa pequeña comercializadora de enceres para niños, siendo el control del inventario el área sensible del negocio, pues está asociado a las ventas, gestión de clientes y artículos. La empresa requiere un sistema de información WEB para el control de sus artículos, su existencia, gestión de clientes y control de ventas.

En la actualidad Chacaticos realiza el control de sus artículos y clientes manualmente a través de hojas de Excel. Su deseo de automatización obedece al crecimiento de la empresa y en querer agilizar y controlar sus operaciones. De este modo se requiere que el sistema realice las siguientes funciones:

- Gestión de Clientes: se requiere la gestión de clientes, cuyos datos de interés son: Nombre, Apellido, Cédula o Rif, dirección, teléfonos, sexo, estado civil, y correo electrónico.
- Gestión de Artículos: se necesita tener registrados los artículos que se comercializan, para ello se requieren los siguientes datos: código, nombre, descripción, fotografía, categoría, costo, precio de venta y existencia.
- Control de Ventas: se requiere registrar las venta de los artículos que la empresa ofrece, para ello es necesario registrar; el cliente que realiza la operación, la fecha, el o los artículos vendidos, y el monto total.

- Consulta Reporte de Ventas: se desea consultar las ventas realizadas, permitiendo seleccionar si la consulta se requiere por fechas y/o cliente.

Una vez que se tienen los requisitos funcionales de la empresa, se comienza con el proceso de la Ingeniería de Requisitos. Se realizan los subprocesos de elicitación y de requisitos, análisis y especificación, donde una vez llegado al subproceso de validación de requisitos, se procede a aplicar el método VAREME, propuesto en la presente investigación. Para aplicar el método VAREME, se deben seguir los siguientes pasos:

1. **Revisión Documento de Especificación de Requisitos**, partiendo que en los subprocesos anteriores de análisis y especificación de requisitos se creó el documento de especificación de requisitos compuestos por los diagramas UML, en este paso se procede hacer su revisión técnica.
   - **Validar la estructura del documento de especificación de requisitos**, en este caso antes de proceder a la validación de los requisitos a través del prototipo se debe validar la estructura del documento de especificación de requisitos, revisando que los modelos en UML que lo conforman estén completos y sean consistente entre ello.

   - **Revisar especificación técnica de los modelos que conforman el documento de especificación de requisitos**, en este paso se revisa con detalle cada uno de los modelos, diagramas en UML que conforman el documento y se realizan las correcciones necesarias en el caso que lo ameriten. Para el ejemplo presentado el modelo de casos de uso, diagrama de clases, diagrama de actividades y modelo de navegación son revisados, obteniendo como salida los siguientes diagramas ya verificados que se muestran a continuación:

a. Modelo de Casos de Uso

Actores:

Figura 31. Actores del Sistema
Fuente: La Autora (2012)

Diagramas de Casos de Uso:

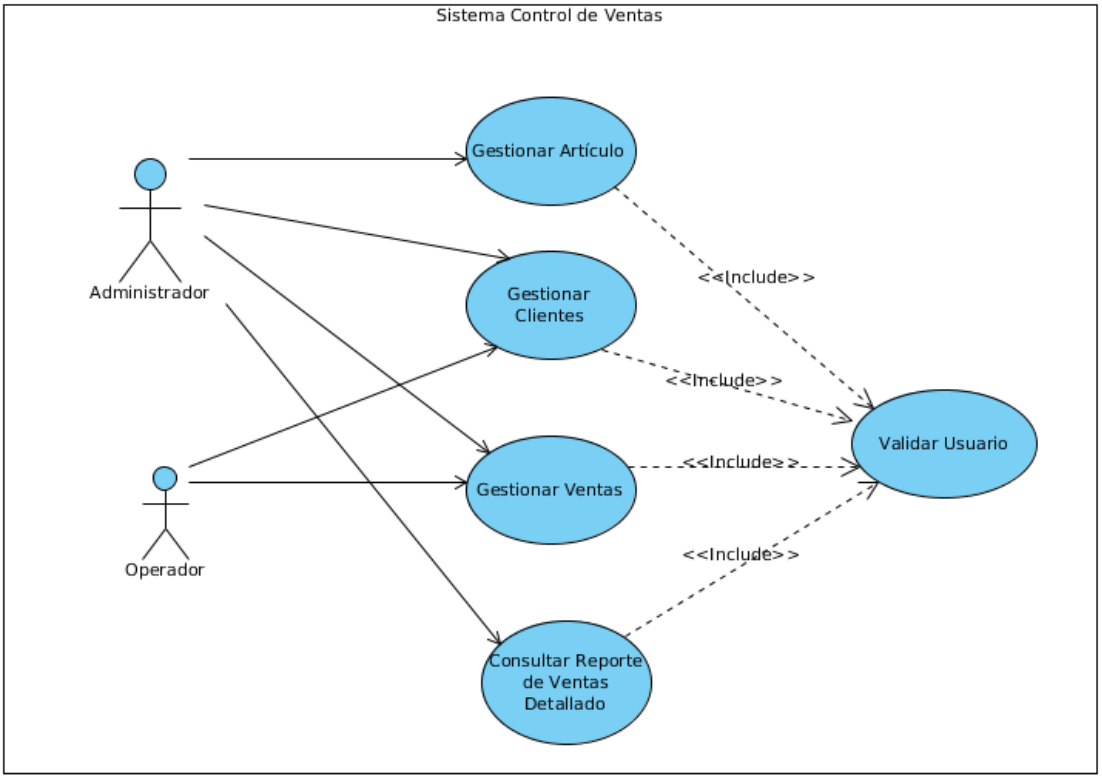

Figura 32. Diagrama Casos de Uso Sistema Control de Ventas
Fuente: La Autora (2012)

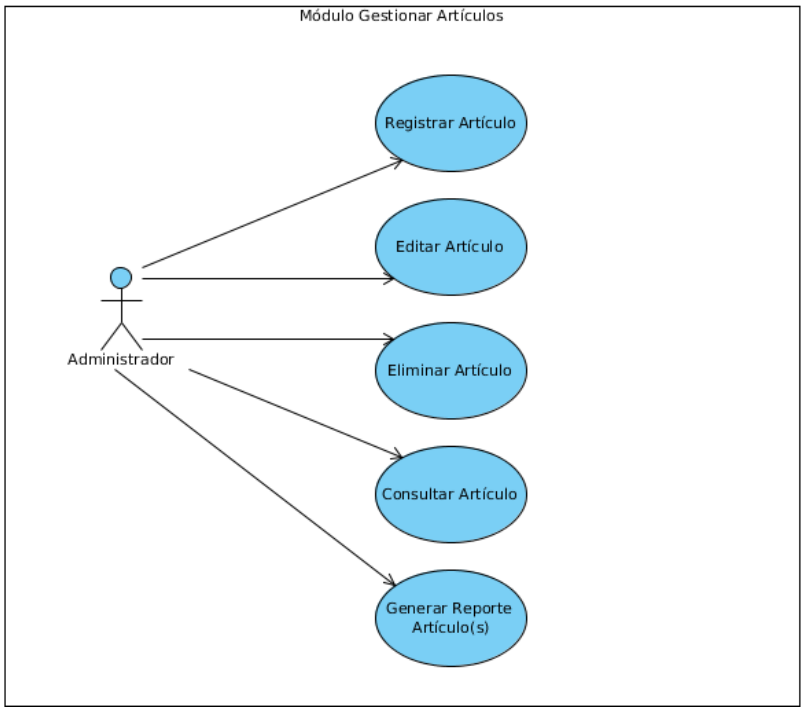

Figura 33. Diagrama Casos de Uso Módulo Gestionar Artículos
Fuente: La Autora (2012)

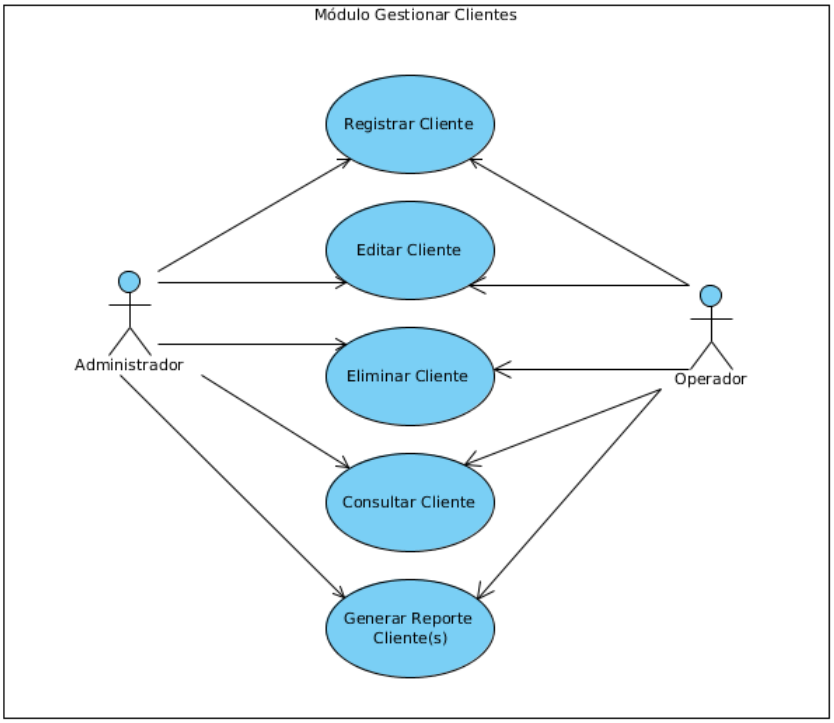

Figura 34. Diagrama Casos de Uso Módulo Gestionar Clientes
Fuente: La Autora (2012)

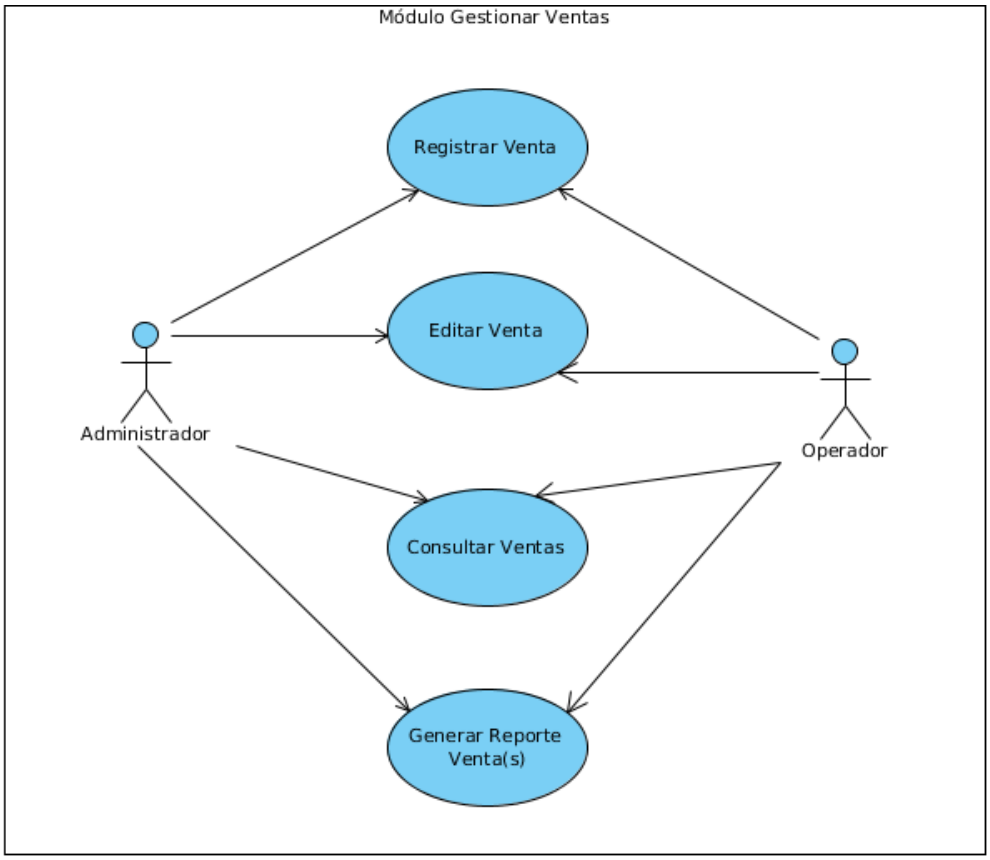

Figura 35. Diagrama Casos de Uso Módulo Gestionar Ventas
Fuente: La Autora (2012)

Descripción Textual Casos de Uso:

Según Montilva y otros (2008), la descripción textual de los casos de uso se aplica a aquellos casos de uso que sean relevantes, complejos o que su flujo de eventos no sea evidente o intuitivo, es por ello que para el ejemplo presentado en el caso de estudio, se describirá solo el caso de uso Registrar Ventas, ya que es el que presenta mayor complejidad. Dicho caso de uso se describe a continuación:

Tabla 27. Descripción Caso de Uso Registrar Venta

Caso de Uso	Registrar Venta		Ref.: 01
Actor(es)	Administrador, Operador		
Descripción	Este caso de uso de uso permite registrar la venta de uno o más artículos para un cliente determinado.		
Precondición	El actor debe haber ingresado al sistema. Los artículos deben estar previamente registrados		
	Paso	***Acción***	
Flujo de Evento Normal	1	Este caso comienza cuando se selecciona la opción Registrar Venta.	
	2	Se hace la búsqueda del cliente, para ello se introduce en el sistema la identificación del mismo, para verificar que se encuentre registrado.	
	3	Se seleccionan el artículo a incluir en la venta, a partir de una lista de los artículos registrados.	
	4	Se introduce la cantidad del artículo.	
	5	El sistema verifica la disponibilidad (stock) del artículo para la cantidad solicitada	
	6	Se repiten los pasos 3 y 5 hasta que se terminen de incluir los artículos en la venta	
	7	El sistema calcula la cantidad total de la venta.	
	8	El Sistema registra la venta y actualiza el stock del artículo.	
	9	Se genera e imprime el recibo de venta.	
Postcondición	Se registra la venta, su importe y su respectivo impuesto. Se actualiza el stock del artículo.		
Flujo de Evento Alternativo	2	Si el cliente no existe, se procede a registrarlo en el sistema.	
	5	En caso que la cantidad de artículo no esté disponible, el sistema emite un mensaje de información y no permite registrar el artículo en la venta con la cantidad indicada.	

Fuente: La Autora (2012)

b. Diagramas de Actividad

Para el presente caso de estudio se especifica sólo el diagrama de actividad del escenario Registrar Ventas, ya que, como se mencionó anteriormente, es el que presentan mayor complejidad.

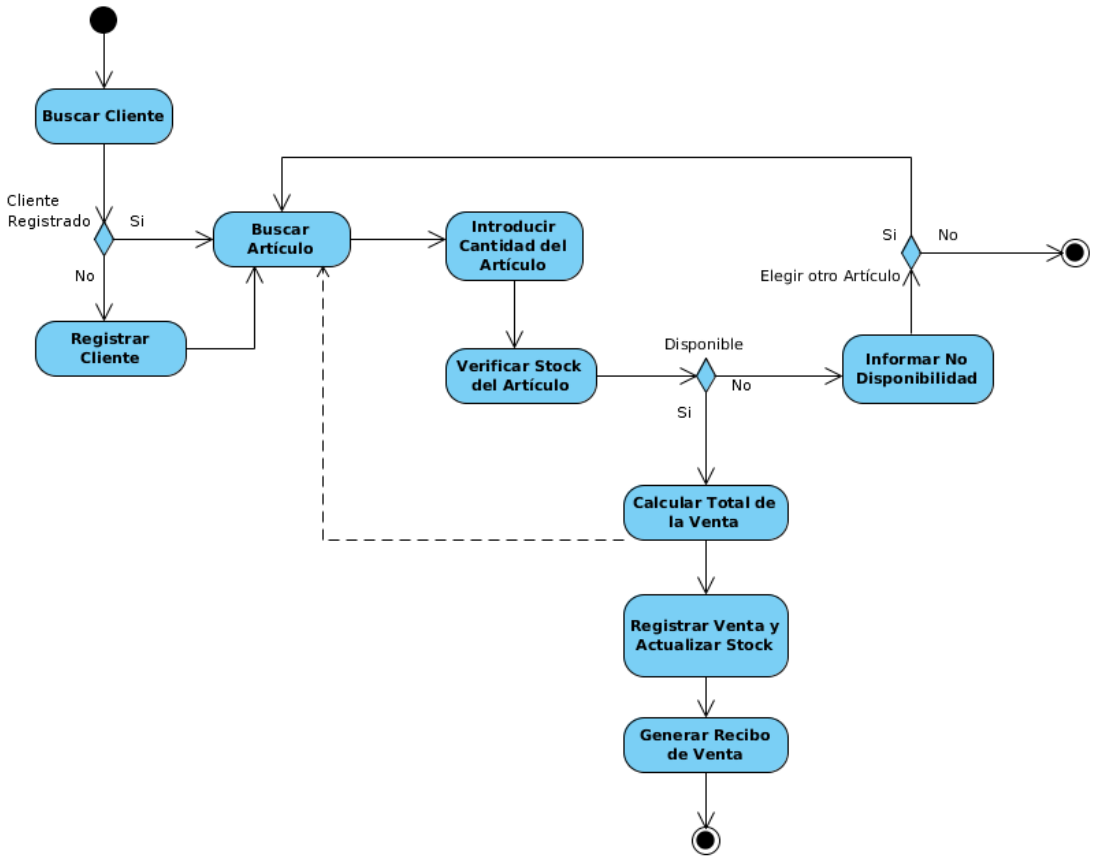

Figura 36. Diagrama de Actividades del Escenario Registrar Ventas
Fuente: La Autora (2012)

c. Diagrama de Clases

A continuación se presenta el diagrama de clases para el caso de estudio presentado:

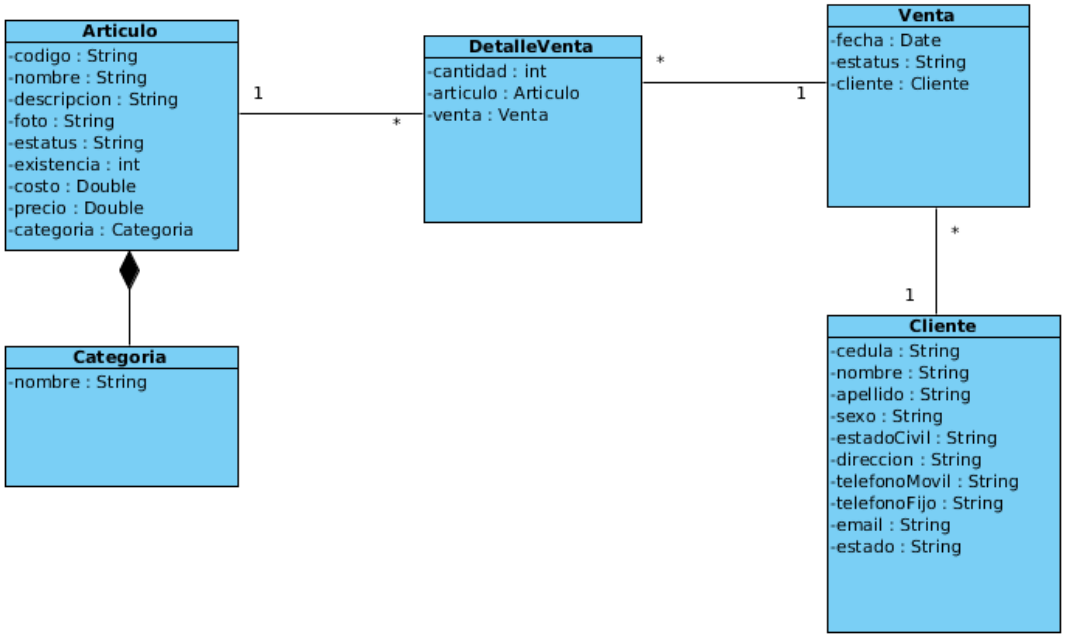

Figura 37. Diagrama de Clases del Sistema Control de Ventas
Fuente: La Autora (2012)

d. Modelo de Navegación

En un sistema para la web es útil saber cómo están enlazadas las páginas. Ello significa que necesitamos un diagrama conteniendo nodos (unidades de navegación) conectadas a través de enlaces, donde los nodos pueden ser representados en diferentes páginas o en una misma página. Para el caso de este ejemplo el modelo de navegación está representado como un diagrama de clases en UML siguiendo la nomenclatura de UWE (UML-based Web Engineering), donde se especifican las siguientes clases de navegación:

- Nodos: punto de la navegación en la que el usuario puede trabajar con la información. Representado con el estereotipo << NO >>.
- Queries (consultas): puntos de la navegación donde el sistema solicita información al usuario que es esencial para continuar con la navegación. Representado con el estereotipo << QU >>.
- Índices: puntos de navegación donde al usuario se le facilita una lista de

posibles resultados a visualizar, todos referidos a la misma información. Representado con el estereotipo << IN >>.

- Menú: punto de la navegación desde el cual el usuario puede ir a varias opciones diferentes. Representado con el estereotipo << ME >>.
- Enlace: cualquier posibilidad de navegación desde una clase navegacional a otra.

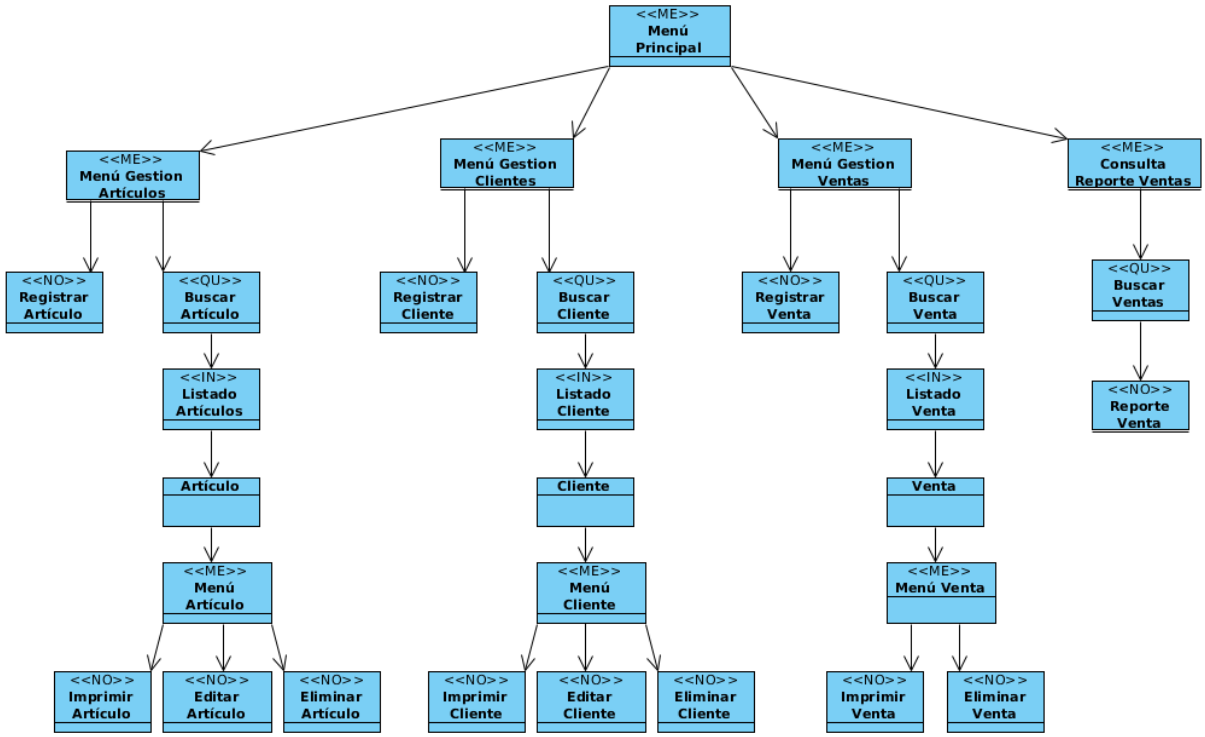

Figura 38. Modelo de Navegación del Sistema Control de Ventas
Fuente: La Autora (2012)

2. Generación Prototipo de Interfaces de Usuario de Aplicaciones Web

- **Generar modelo lógico de datos**, el modelo lógico de datos ayuda a proveer la funcionalidad del prototipo. Para este caso de estudio se utilizó una base de datos relacional como modelo lógico de datos. Existen herramientas de software que permiten generar la base de datos relacionales de forma automática

partiendo de las clases definidas en el diagrama de clases. Para el ejemplo presentado en este caso de estudio, a través de una funcionalidad del software Visual Paradigm (herramienta case que se usó para modelar los diagramas UML), se generaron las tablas con sus campos y claves foráneas, partiendo de las clases, sus atributos y relaciones definidas en el diagrama de clases. Para ello, primero a partir del Diagrama de Clases se genero el Diagrama Entidad Relación, y luego en base al mismo se crearon las tablas en forma automática, seleccionando como gestor de base de datos, para este ejemplo, MySQL.

A continuación se muestra el diagrama entidad relación generado a partir del diagrama de clases previamente definido:

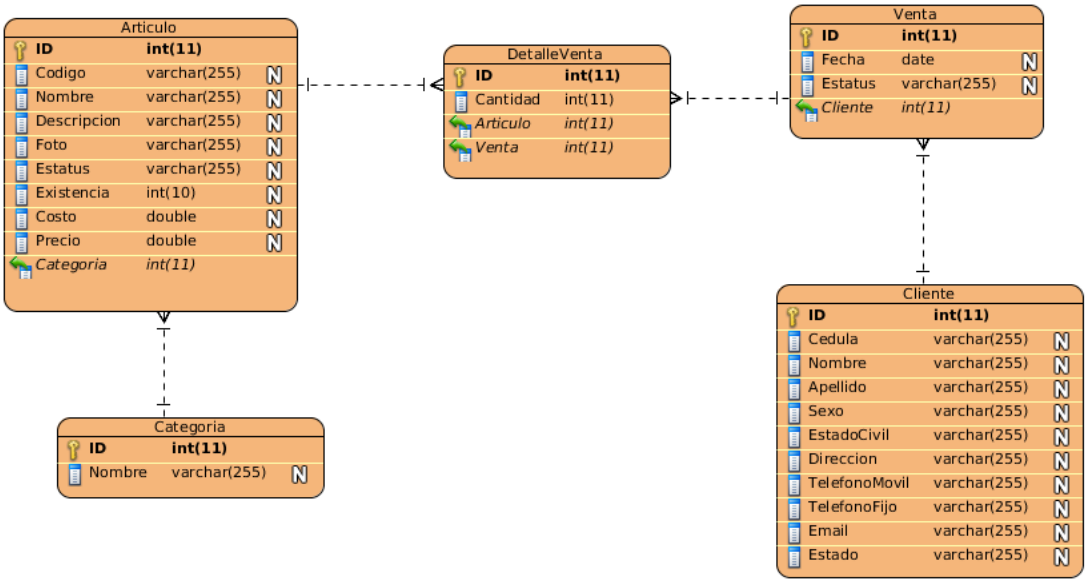

Figura 39. Diagrama Entidad Relación del Sistema Control de Ventas
Fuente: La Autora (2012)

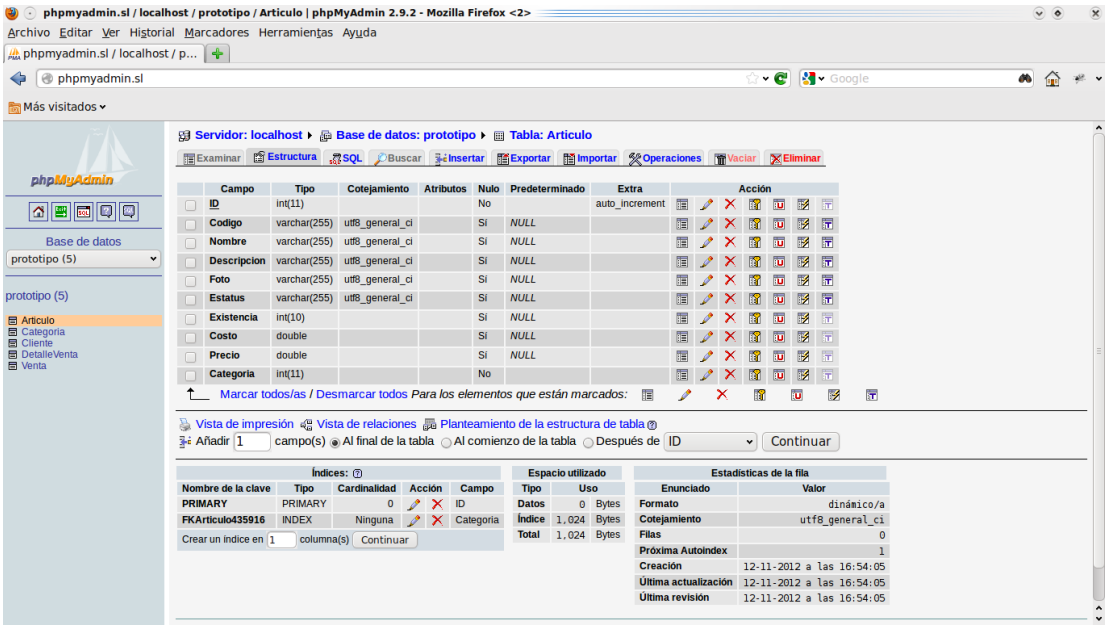

Figura 40. Base de datos relacional generada en MySQL del Sistema Control de Ventas
Fuente: La Autora (2012)

- **Generar Código y Páginas Web**, para llevar a cabo el desarrollo del prototipo para el caso de estudio propuesto y que este basado en patrones RIA, se seleccionó como herramienta PHP Runner, la cual es una aplicación capaz de generar automáticamente todo un repertorio de páginas web para insertar información en los campos de una base de datos determinada y soporta múltiples controles (widgets) que permiten aplicar patrones RIA. El lenguaje empleado en estas páginas es PHP. El proceso de creación de las páginas se realiza completando un asistente, en el cual partiendo de una base de datos hay que decidir los campos que se mostrarán en las páginas, los usuarios que podrán añadir contenido, las funciones que estarán disponibles, entre otros.

  Con lo cual basado en la base de datos que fue previamente generada a partir del diagrama de clases, y tomando como referencia los diagramas de caso de uso, de actividades y el modelo de navegación, se fue generando el prototipo de forma semi-automática con el PHP Runner. También se tomó como base el documento de Patrones RIA recomendados a usar, (ver tabla 23) para incorporar los

patrones RIA que se consideren necesarios según las funcionalidades planteadas. A continuación se muestran las pantallas principales del prototipo generado para el caso de estudio presentado, con algunos datos de prueba:

Figura 41. Pantalla de Autenticación de inicio al prototipo de la aplicación
Fuente: Autor de la Investigación

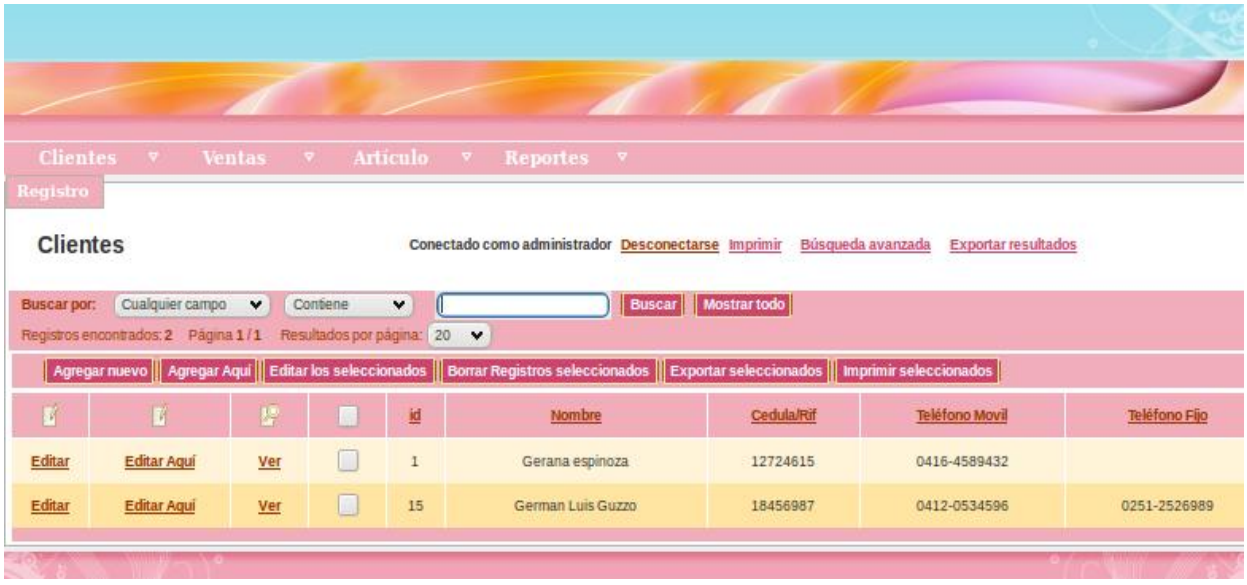

Figura 42. Gestión de Clientes. Pantalla inicial (listado)
Fuente: Autor de la Investigación

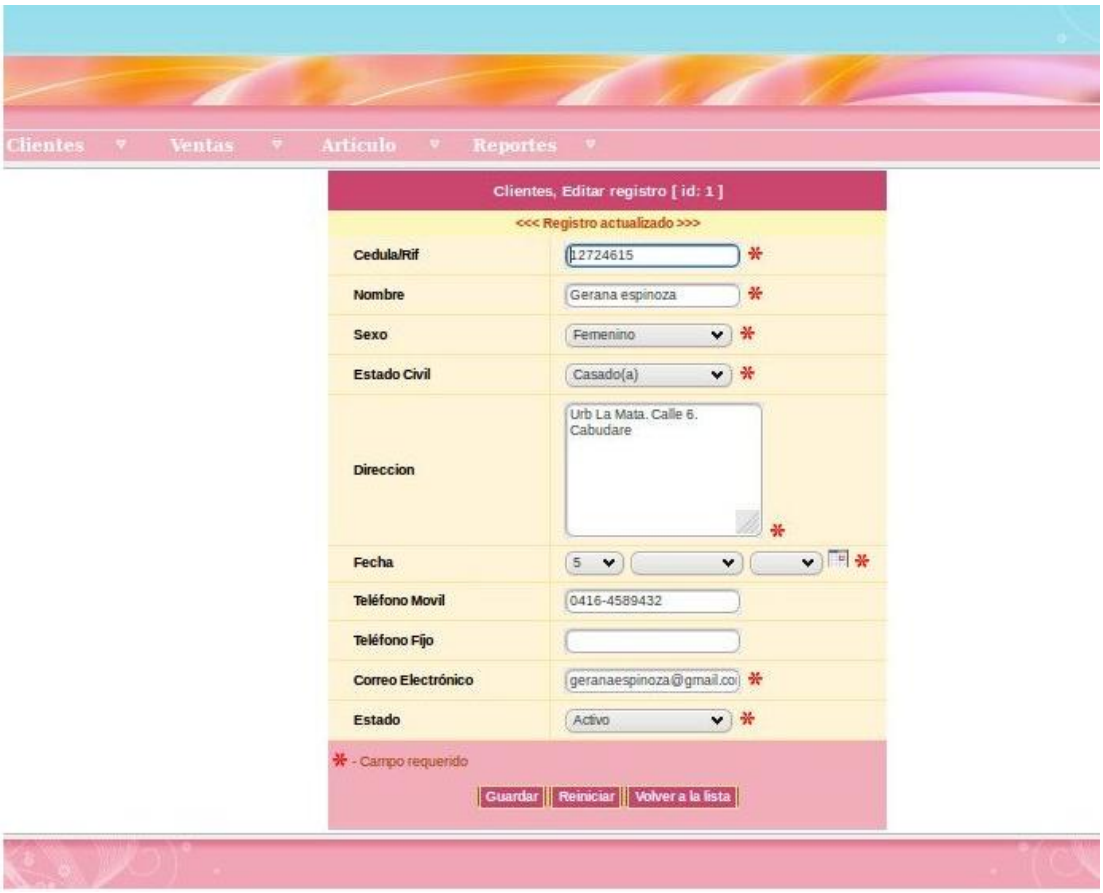

Figura 43. Pantalla de Edición de Clientes
Fuente: Autor de la Investigación

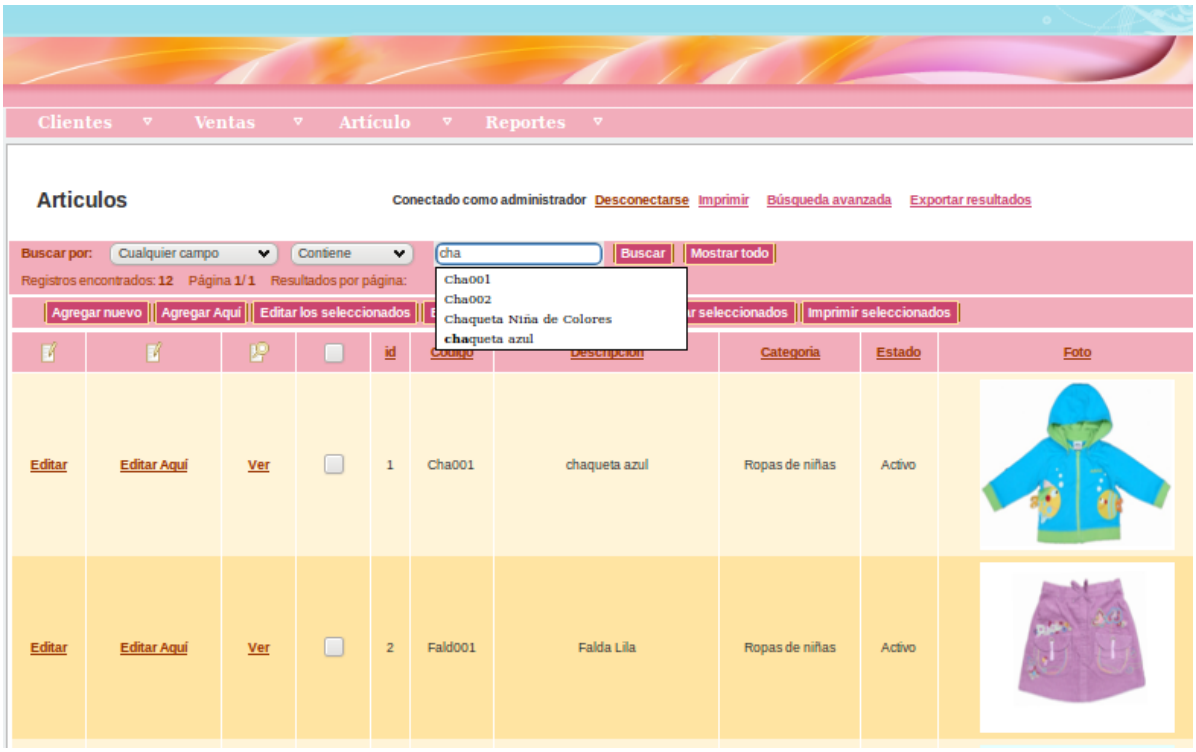

Figura 44. Gestión de Artículos. Pantalla inicial (listado)
Fuente: Autor de la Investigación

Figura 45. Gestión de Ventas. Pantalla inicial (listado)
Fuente: Autor de la Investigación

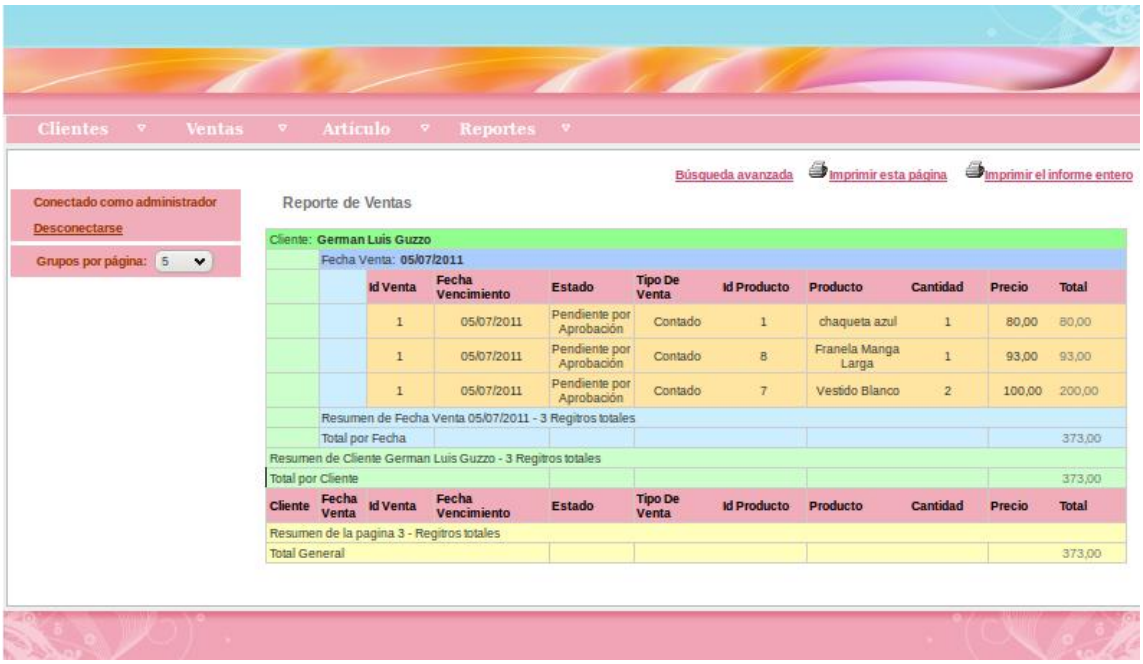

Figura 46. Pantalla Reporte de Ventas Realizadas
Fuente: Autor de la Investigación

- **Evaluar los patrones RIA del prototipo de interfaces de usuario, por parte del ingeniero de software,** una vez generado el prototipo, se recomienda revisar los patrones RIA incorporados en el mismo. Pare ello el ingeniero de software debe evaluar el prototipo generado a través del cuestionario propuesto (ver tabla 24), con el fin de comprobar si el prototipo cumple con los requisitos mínimos necesarios para que pueda ser considerado un prototipo basado en patrones RIA y así poder continuar con el flujo de actividades del método VAREME.

  A continuación se muestra el cuestionario respondido por el ingeniero de software aplicado al prototipo generado para el caso de estudio presentado.

Tabla 28. Cuestionario de Evaluación del Uso de Patrones RIA aplicado al Prototipo del Caso de Estudio presentado

Preguntas		Valoración	
		SI	NO
1	¿Uso del control calendario para las Fechas?	X	
2	¿Uso de control con mascaras que den formato?		X
3	¿Uso de herramientas que se revelan con el pase del ratón (mouse hover)?		X
4	¿Uso de herramientas que se revelan u ocultan con una acción?	X	
5	¿Uso de Menú secundario?	X	
6	¿Proveer atajos de teclado (combinación de teclas) para realizar una acción?	X	
7	¿Se permite la selección alterna de elementos?		X
8	¿Se permite la selección de elementos usando checkbox?	X	
9	¿Se puede arratrar y soltar elementos (Drag and drop)?		X
10	¿Uso de ToolTipText?	X	
11	¿Uso de Overlay: superponer los mensajes de ayuda o ilustraciones directamente sobre la interfaz?	X	
12	¿Uso de menús de navegación?	X	
13	¿Uso de menú Breadcrumbs (migas de pan)?		X
14	¿Vista previa activa?	X	
15	¿Uso de indicador de progreso?		X
16	¿Actualización periódica de los datos?	X	
17	¿Mensaje retroalimentación cada vez que se ejecuta una acción?	X	
18	¿Uso de tabla (Grid) para presentar los datos?	X	
19	¿ Selección e interacción con datos de la tabla (grid)?	X	
20	¿Uso de paginación en la tabla (grid) de datos?	X	
21	¿Uso de filtros o búsquedas en la tabla (grid) de datos?	X	
22	¿Editando registros directamente en la tabla (grid)?	X	
23	¿Uso del control autocompletar?	X	
24	¿Búsquedas avanzadas?	X	

Fuente: La Autora (2012)

Evaluando el resultado obtenido del cuestionario, de un total de 24 preguntas se tienen 18 respuestas afirmativas y 6 respuestas negativas, con lo cual el porcentaje de respuestas afirmativas resultante es de 75%, por lo tanto según el criterio definido para evaluar el resultado de este cuestionario se concluye que el prototipo generado cumple con los patrones RIA necesarios para ser considerado un prototipo basado en

patrones RIA.

Algunos de estos patrones RIA los podemos observar en las pantallas mostradas en las figuras anteriores, por ejemplo el uso de menú de navegación, uso de tabla (grid) para presentar los datos, selección e interacción con datos de la tabla, uso de paginación en la tabla (grid) de datos, uso de filtros o búsquedas en la tabla (grid) de datos, se permite la selección de elementos usando checkbox, se pueden eliminar varios elementos al mismo tiempo seleccionándolos todos. En la figura 33 se muestra el control calendario para las fecha. En la figura 34 se muestra el uso del control autocompletar, y el uso de las búsquedas avanzadas. En la figura 35 se presenta el uso de overlay, superponer los mensajes de ayuda o ilustraciones directamente sobre la interfaz

3. Validación de Requisitos a través del Prototipo de interfaces de usuarios basado en Patrones RIA

- **Establecer sesiones de validación según los escenarios y roles**, para este ejemplo existen dos roles o actores definidos, el administrador que tiene acceso a todas las opciones del sistema, y el usuario operador, quien se encargará de registrar las ventas y podrá incluir clientes en el caso que no existan cuando se procese una venta de artículos.

  Con lo cual según estos dos roles definidos se definen dos sesiones de validación, una donde se involucran al o los usuarios que vayan asumir el rol de administrador, y otra sesión con los usuarios que desempeñaran el rol de operadores. El ingeniero de software debe ponerse de acuerdo con el cliente para definir el lugar, fecha y hora de estas sesiones, donde el cliente decidirá que usuarios participarán en cada sesión según los roles definidos, este acuerdo se estable por escrito en un documento de establecimiento de sesiones de validación (calendario de reuniones), de la siguiente manera:

Sesiones De Validación del Prototipo Establecidas al 14 de Noviembre del 2012

Sesión Nro. 1		
Fecha: 05/12/2012	**Hora:** 3:00 pm	**Lugar:** Sala de Reuniones de la Empresa Chacaticos.
Validación del Prototipo por parte de los usuarios con el rol Administrador		
Reunión convocada por:	Ingeniero de Software	
Asistentes:	Usuario Administrador 1 Usuario Administrador 2	
Responsable:	Usuario Administrador 1	
Observaciones:		
Sesión Nro. 2		
Fecha: 07/12/2012	**Hora:** 3:00 pm	**Lugar:** Sala de Reuniones de la Empresa Chacaticos.
Validación del Prototipo por parte de los usuarios con el rol Operador		
Reunión convocada por:	Ingeniero de Software	
Asistentes:	Usuario Operador 1 Usuario Operador 2 Usuario Operador 3 Usuario Operador 4	
Responsable:	Usuario Operador 3	
Observaciones:		

Figura 47. Documento de Sesiones de Validación Establecidas
Fuente: Autor de la Investigación

- **Realizar sesiones de validación,** según las sesiones de validación definidas y establecidas en el documento previamente especificado, se llevan a cabo las sesiones de validación correspondientes, las cuales consiste en una reunión entre el ingeniero de software y los usuarios finales con el objetivo que estos interactúen con el prototipo de la aplicación, apoyándose los diagramas que conforman el documento de especificación de requisitos, para validar los requisitos definidos.

  Así, una vez que se hayan realizado las sesiones de validación, según la retroalimentación obtenida por parte de los usuarios finales en cuanto a que existan requisitos nuevos descubiertos y/o se corrijan errores u omisiones importantes, es necesario entonces ajustar los modelos del documento de especificación de requisitos para incorporar estos cambios y realizar nuevamente las sesiones de validación con los usuarios finales.

4. **Ajuste Documento de Especificación de Requisitos**
   - **Modificar modelos y descripciones:** según la retroalimentación proporcionada en durante las sesiones de validación, las descripciones textuales de los requisitos, así como los diagramas de casos de uso y diagramas de clase elaborados para el presente ejemplo, deben adaptarse para incorporar las correcciones correspondientes.
   - **Verificar consistencia e integridad de la especificación técnica:** luego de realizados las correcciones pertinentes en los diagramas UML correspondientes se verifica su consistencia e integridad entre ellos.

5. **Definición de Parámetros de Aceptación de la Aplicación**
   - **Evaluar la calidad del prototipo de la aplicación a través de un cuestionario, por parte de los usuarios finales**: una vez finalizadas las sesiones de validación, luego que usuarios haya interactuado con el prototipo, se

procede evaluar la calidad del prototipo por parte de los usuarios a partir del **Instrumento de Evaluación de la Calidad del prototipo (ver tabla 22),** el cual es un cuestionario que consta de 62 preguntas cerradas, donde cada una consta de 5 opciones posibles de respuesta.

Para el caso propuesto el cuestionario se aplicó seis (6) usuarios de la empresa Chacaticos quienes serán los encargados de usar el sistema. Dos (2) de ellos con roles de usuario administrador y los otros cuatro (4) como usuario operador, dichos usuarios interactuaron con el prototipo en las sesiones de validación. Los resultados obtenidos luego de ser aplicado el cuestionario de evaluación de calidad a cada uno de ellos fue el siguiente:

Usuario Evaluado	Total Puntaje Obtenido en el Cuestionario
Usuario Administrador 1	285
Usuario Administrador 2	255
Usuario Operador 1	300
Usuario Operador 2	270
Usuario Operador 3	290
Usuario Operador 4	285

Si se promedia el resultado obtenido tendremos como valor resultante de 280 puntos. Para analizar este valor resultante es importante establecer el rango mínimo y máximo para esta escala de medición:

- El máximo puntaje que se puede obtener, es que todas las respuestas sean respondidas como "Totalmente Cumplido", el cual corresponde a 310 puntos (62 preguntas x 5 puntos).
- El mínimo puntaje que se pude obtener es que todas las respuestas sean respondidas como "Incumplidas", el cual correspondería a un valor de 62 puntos (62 preguntas por 5 puntos)

Por lo tanto el valor resultante obtenido después de aplicar el cuestionario a los ocho (8) usuarios mencionados, corresponde a 280 puntos con lo cual se nota claramente que se acerca al Puntaje Máximo.

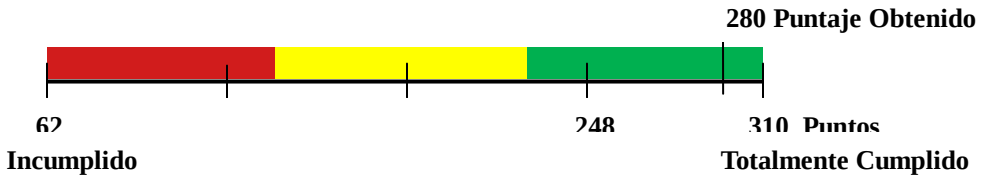

Figura 48. Gráfico escala de likert del cuestionario aplicado
Fuente: La Autora (2012)

Con lo cual se puede concluir que el valor se acerca a "Totalmente Cumplido", lo cual demuestra que el prototipo presentado cumple con los factores de calidad evaluados, lo cual ayudará a garantizar la calidad del producto final y la satisfacción del usuario.

- **Determinar parámetros de aceptación de la aplicación:** se definen los parámetros de aceptación a utilizar para lograr un acuerdo con los usuarios finales luego de haber validado los requisitos a través del prototipo.

- **Verificar los parámetros de aceptación**: verificar, con los interesados, los parámetros de aceptación de la aplicación y una vez llegado a un acuerdo generar el documento de aceptación de requisitos y culminar el proceso de validación. El documento de aceptación de requisitos para el caso de estudio presentado es el siguiente:

Empresa Desarrolladora de Software S.A.
Departamento de Ingeniería de Requisitos

Documento de Aceptación de Requisitos

Señores Empresa Chacaticos
Ciudad.

Por medio de la presente se hace constar el común entendimiento, la aceptación y conformidad de los requisitos definidos, especificados y validados para el proyecto en curso, se anexan los diagramas revisados y las pantallas del prototipo evaluado.

Las futuros modificaciones que se requieran, se realizarán de acuerdo al procedimiento de cambios definidos para el proyecto, entendiendo que los cambios aprobados podrán requerir que se renegocien los acuerdos actuales sobre costos, recursos asignados y fecha de entrega del proyecto.
Sin más que agregar,

Empresa Desarrolladora de Software S.A. Empresa Chacaticos

Figura 49. Documento de Aceptación de Requisitos
Fuente: Autor de la Investigación

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

El presente capítulo tiene como finalidad presentar las conclusiones y aportes que se establecieron al desarrollar el presente trabajo de investigación, así como las recomendaciones para futuras investigaciones. Las conclusiones se detallan a continuación:

- El presente estudio se centró en diseñar, construir y validar un método de validación de requisitos de software a partir de prototipos de interfaces de usuario basados en patrones RIA, bajo un enfoque de calidad, el cual fue denominado VAREME.
- En el método VAREME se destaca la importancia de la participación de los usuarios finales en el proceso de validación de requisitos. Es por ello que en aplicación del método los usuarios finales interactúan con un prototipo funcional utilizado iterativamente para validar requisitos, de este modo el prototipo evoluciona hasta que los usuarios finales compartan un entendimiento común con los ingenieros de software.
- Para construir el método VAREME se partió del enfoque basado en ensamblaje de la Ingeniería de Métodos Situacional de Weerd y otros (2006), lo cual permitió conocer con detalle los pasos a seguir para construir un nuevo método, tomando fragmentos de otros métodos existentes, para ello se estudiaron trabajos en el área de validación de requisitos, a través de prototipo de interfaces de usuario, de los cuales fueron seleccionados tres (3), el Método Gray Watch de Montilva y otros (2008), el método de Análisis y Validación de

Requisitos a través de la Generación Automática de Prototipos de Interfaces de Usuario para Aplicaciones Web, de Ogata y Matsuura (2010) y la metodología diseño del pensamiento de Gabrysiak y otros (2011). Estos trabajos se convirtieron en los métodos candidatos de los cuales se tomaron los fragmentos relevantes y necesarios para construir el método propuesto en la presente investigación en base a los objetivos planteados.

- Se utilizó la técnica de meta-modelado definida por Weerd y Brinkkemper (2009) para representar, de cada uno de los métodos candidatos así como del método VAREME, las actividades y productos entregables de cada actividad en un único diagrama denominado Process Deliverable Diagram, PDD; el cual consiste en dos diagramas integrados, donde la vista de procesos se basa en un diagrama de actividad UML, y la vista de productos se basa en un diagrama de clases UML.
- Se abordó de una manera detallada la tecnología relacionada con los patrones RIA existentes y más utilizados en el desarrollo de aplicaciones Web, de Scott y Neil (2009), los cuales en la actualidad responden a las exigencias de los usuarios finales en materia de interacción humano computador y usabilidad. Esto sirvió como elemento base para la creación del prototipo de interfaz, los cuales garantizan interfaces interactivas y amigables que permiten satisfacer las necesidades y expectativas de los usuarios.
- Para incorporar un enfoque de calidad al método VAREME, se elaboró un cuestionario para ser aplicado a los usuarios finales para así obtener una retroalimentación del proceso realizado. El cuestionario se fundamenta en la definición de un conjunto de métricas, para ello se parte del estudio realizado por Macías y Gómez (2010), tomando los factores de calidad definidos en su trabajo seleccionados de los modelos McCall e ISO 9126-1. Estos factores se adaptaron al modelo de calidad ISO/IEC 25010, en base a las nuevas características y sub-características que este modelo incorpora.

- Para definir las métricas de calidad para la característica usabilidad, se tomó como basé en el trabajo de Fernandez (2009), el cual parte de un modelo para productos de software genéricos, con una extensión a productos orientados a la Web basado en los principios heurísticos de Nielsen (2005) y a la norma ISO/IEC 25010. El introducir la perspectiva de usabilidad permite que cualquier sistema interactivo pueda ser utilizado con efectividad, eficiencia y satisfacción por los usuarios finales.
- El método VAREME permite garantizar la validación de requisitos de software, donde el usuario final comienza a experimentar la interacción con un elemento dinámico: el prototipo de interfaces de usuario basado en patrones RIA, que le ayuda a ir refinando sus exigencias y expectativas para el sistema que está naciendo. Donde la participación del usuario final en un proceso iterativo e incremental ayuda a mejorar la comunicación entre el ingeniero de software y los usuarios, estableciendo una mejor concordancia entre sus necesidades y el sistema, asegurando así el éxito en la Ingeniería de Requisitos.

A partir del método propuesto se obtuvieron los siguientes aportes de la investigación:

- Se realizó una guía con los patrones RIA recomendados a incorporar en el prototipo de interfaces de usuario Web, la cual debe ser utilizada por el ingeniero de software en el momento de generar el prototipo de interfaces de usuario.
- Se elaboró un cuestionario, el cual debe ser aplicado al ingeniero de software, con el fin de evaluar si el prototipo generado cumple con los requisitos mínimos necesarios para que pueda ser considerado un prototipo basado en patrones RIA. Este cuestionario está conformado por un conjunto de preguntas cerradas con dos (2) opciones de respuesta (Si ó No).
- Se diseño y elaboró un instrumento para evaluar la calidad del prototipo de interfaces de usuario basado en patrones RIA del método VAREME, en base a

los factores de calidad del modelo ISO/IEC 25010 seleccionados. Este instrumento es un cuestionario, el cual está compuesto por un conjunto de preguntas cerradas que serán valoradas bajo la escala de likert. Este cuestionario será aplicado a los usuarios finales que intervengan en el proceso de validación de requisitos, una vez que hayan interactuado con el prototipo, con el fin de de mejorar la calidad del producto final y satisfacer las necesidades del usuario.

Así, basándose en la investigación realizada se presentan las siguientes recomendaciones y trabajos futuros:

- Verificar la correctitud y validez del método VAREME a través de su evaluación por parte de expertos en el área de Ingeniería de Requisitos, con el fin de obtener mejoras en el mismo.
- Validar el método VAREME a través de su uso en un caso de estudio real llevándolo a la práctica con un grupo de usuarios e ingenieros de software.
- Continuar con la aplicación del resto de las etapas del método de investigación para la Ingeniería de Software propuesto por Marcos y Marcos (1998), utilizado en la presente investigación, del cual solo se siguieron y completaron las primeras cuatro etapas.
- Usar el método VAREME para ofrecer a la comunidad de ingeniería de software (empresas privada, grupos organizados, el área académica tanto pregrado como postgrado, entre otros), un marco de referencia en lo que respecta al proceso de validación de requisitos de software.
- Asimismo, se considera oportuno recomendar estudios futuros que surjan a partir de la problemática aquí especificada que puedan impulsar nuevos proyectos en la comunidad científica.

REFERENCIAS BIBLIOGRÁFICAS

Abrahão, S. y Insfran, E. (2006). *Early Usability Evaluation in Model-Driven Architecture Environments.* 6th IEEE International Conference on Quality Software (QSIC), Beijing, China. pp. 287-294.

Anaya, R., Londoño, L. y Hurtado, J. (2006). *Una estrategia para Elevar la Competitividad de las Industrias de Software PYMES*. IX Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software - IDEAS. La Plata, Argentina.

Aguilar, J., Garrigós I., Casteleyn, S. y Mazón J. (2011). *Una propuesta orientada a objetivos para el análisis de requisitos en RIA*. XVI Jornadas de Ingeniería del Software y Bases de Datos. JISBD 2011. Coruña, España, pp. 211-224

Allaire, J. (2002) *Macromedia Flash MX: A Next Generation Rich Client, Macromedia White Paper*. [Documento en línea]. Disponible: http://www.c2isoft.in/white-papers/richclient.pdf [Consulta: 2012, mayo 20].

Arias, M. (2005). La Ingeniería de Requisitos y su Importancia en el Desarrollo de Proyectos de Software. *Revista InterSedes Universidad de Costa Rica*, 4 (10).

Aouad, G., y Arayici, Y. (2010). *Requirements Engineering for Computer Integrated Environments in Construction*, Oxford, USA: Wiley-Blackwell.

Berenbach, B., Paulish, D., Kazmeier, J. y Rudorfer, A. (2009). *Software & Systems Requirements Engineering: In Practice*. New York, USA: McGraw-Hill.

Bohem, B. (1981). *Software Engineering Economic*. U.S.A.: Prentice-Hall.

Brinkkemper, S. (1996). *Method engineering: engineering of information systems development methods and tools. University of Twente. Information and Software Technology*. [Artículo en linea]. Disponible: http://doc.utwente.nl/18012/1/Brinkkemper96method.pdf [Consulta: 2012, junio 01]

Bunge, M. (1983), La Investigación Científica. Barcelona: Ariel.

Blanco, S. (2011). *Desarrollo de librería Django para Desarrollo de RIA*. Trabajo de grado de Maestría Administración Web. Universidad Abierta de Cataluña. España.

Buzzo, M. y Rivero, J. (2009), *Definición de Rich Internet Applications a través de*

Modelos de Dominio Específico. Trabajo de grado de Licenciatura en Informática. Universidad Nacional de la Plata. Argentina.

Busch, M., Koch, N. (2009). *Rich Internet Applications. Technical Report 0902*, Institute for Informatics, Ludwig-Maximilians-Universität München, Germany.

Calero, C., Ruiz, J. y Piattini, M. (2005). *Classifying Web metrics using the Web Quality Model*. Emerald Group. Publishing Limited. 29(3), pp. 227-248.

Campana, D. (2009). *Especificación de Interfaces y Patrones RIA*. Trabajo de grado de Maestría en Ingeniería de Software. Universidad Nacional de la Plata. Argentina. Disponible: http://www.sedici.unlp.edu.ar?id=ARG-UNLP-TPG-0000000719 [Consulta: 2012, febrero 19]

Callaos, N. y Callaos, B. (1993). *Designing with Systemic Total Quality.* Orlando: USA. International Conference on Information Systems, International Institute of Informatics and Systemics.

Christel, M., y Kang, k. (1992) *Issues in Requirements Elicitation*, U.S.A. Software Engineering Institute.

Codd, E. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM . 13(6), pp. 377-387.

Covella J. (2005). *Medición y Evaluación de Calidad en Uso de Aplicaciones Web*. Trabajo de Grado de Maestría en Ingeniería de Software. Universidad Nacional de la Plata. Argentina

Dolado, J. y Fernández, L. (2000). *Medición para la gestión en la Ingeniería del Software*. Madrid, España: Ra-Ma.

Dominguez, J. (2009). *The Curious Case of the CHAOS Report 2009. Project Smart*. [Documento en línea]. Disponible: http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html. [Consulta: 2012, mayo 20]

Dromey, R. (1998). *Software Product Quality: Theory, Model and Practice.* Australia. Griffith University, Software Quality Institute.

Falkenberg, E., Hesse, W. y Linggreen P. (1996), *A Frameeork of Information System Concepts* – The FRISCO Report. 221 p.

Farré, X. (2005). *Rich Internet Applications.* Trabajo Final de Carrera. Universidad Politécnica de Cataluña, España.

Fernandez, A. (2009). *WUEP: Un Proceso de Evaluación de Usabilidad Web Integrado en el Desarrollo de Software Dirigido por Modelos*. Trabajo de Grado de Maestría. Universidad Politécnica de Valencia, España.

García-Cordoba, F. (2007). *La Investigación Tecnológica*. México: Limusa.

Gabrysiak, G., Giese, H., Seibel, A. (2011), *Towards Next Generation Design Thinking: Scenario-Based Prototyping for Designing Complex Software Systems with Multiple Users. Design Thinking – Understand, Improve, Apply* (pp. 219-236). Berlin, Alemania: Springer.

Garrett, Jesse (2005). *Adaptive Path*. [Página web en línea]. Disponible: http://adaptivepath.com/ideas/essays/archives/000385.php [Consulta: 2012, marzo 15]

Granollers, T. (2004). *MPIu+a. Una metodología que integra la Ingeniería del Software, la Interacción Persona-Ordenador y la Accesibilidad en el contexto de equipos de desarrollo multidisciplinares*. Tesis Doctoral, Universidad de LLeida.

Harmsen, A. (1997). *Situational Method Engineering*. University of Twente. Netherlands.

Halim, S.,Abang, D. y Deris, S. (2011). *Variability Integration in Software Product Line Core Assets.* The 3rd Software Engineering Postgraduate Workshop (SEPoW), pp. 63-68.

Henderson-Sellers, B. y Ralyté, J. (2010). *Situational method engineering: state-of-the-art review.* Journal of Universal Computer Science. 16(3), pp. 424–478.

Hernández, R., Fernández, C. y Baptista, L. (2006). *Metodología de la Investigación.* (4ta. Ed.). México: McGraw-Hill.

Howles, T. (2003). *Widespread Effects of Defects*. Quality progress, 36(8), pp. 58-61.

Hong, S., Goor, G. van den, y Brinkkemper, S. (1993). *A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies*. Proceedings of the 26th Hawaii International Conference on Systems Science. p. 689-698.

IEEE. (1990*). IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. The Institute of Electrical and Electronics Engineers. New York,

USA. Disponible: http://standards.ieee.org/findstds/standard/610.12-1990.html [Consulta: 2012, Febrero 16].

IEEE. (1998). Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998.

ISO/IEC 25010 (2010). *System and Software Engineering – System and Software Product Quality Requirements and Evaluation* (SQuaRE) – System and software quality models.

ISO/IEC 25000 (2005). *Calidad del Producto de Software y la norma* ISO/IEC 25000. [Sitio Web]. Disponible: http://www.iso25000.com/ [Consulta: 2012, julio 20]

ISO/IEC 9126 (1991) *Software Product Evaluation–Quality Characteristic and Guidelines for their Use*. International Standard 9126–1991, International Organization for Standarization. Montreal, Quebec

ISO/IEC 9126-1 (2001). *Software Engineering – Product Quality Part 1*. Quality model.

Ivory, M. y Megraw, R. (2005). *Evolution of Web Site Design Patterns*. ACM Transactions on Information Systems, 23(4), pp 463-497.

Jacobson, I. (1993). *Object-Oriented Software Engineering*, U.S.A.:Addison-Wesley.

Jimenez, A. (2012). *Incorporando la Gestión de la Trazabilidad en un entorno de Desarrollo de Transformaciones de Modelos Dirigido por Modelos*. Tesis Doctoral. Universidad Rey Juan Carlos. Madrid, España.

Joyanes, L. (1996). *Programación Orientada a Objetos* (7a. ed.). Madrid, España: McGraw-Hill.

Kalareh, A. (2012). *Evolving Software Systems for Self-Adaptation.* Tesis Doctoral. Universidad de Waterloo, Canada.

Kendall, K. y Kendall, J. (1995). *Análisis y Diseño de Sistemas* (3a. ed.). Mexico, Prentice Hall Hispanoamericana.

Kirakowski, J. (2004). Likert, and the Mathematical basis of Scales. University College Cork, Irlanda. [Página web en línea]. Disponible: http://www.keysurvey.com/resources/likert_and_the_mathematical_basis_of_scales.jsp [Consulta: 2012, agosto 28]

Koch, N., Pigerl, M., Zhang, G. y Morosova, T. (2009). *Patterns for the Model-based Development of Rich Internet Applications.* 9th International Conference on Web Engineering. (ICWE 2009), LNCS 5648. pp. 283-291.

López De Luise, M., y Agüero, M. (2007). Aplicación de Métricas Categóricas en Sistemas con Lógica Difusa. *IEsatin America Transaction*, 5(1), pp. 55-61.

Lopez M. (2011). *ArchiMeDeS: A Service-Oriented Framework for Model-Driven Development of Software Architectures*. Tesis Doctoral. Universidad Rey Juan Carlos. Madrid, España.

Lujan, S. (2002). *Programación de Aplicaciones Web: Historia, principios básicos y clientes Web*. España: Club Universitario.

Macías, J. y Gómez, J. (2010), *Utilizando el Modelo de Calidad de McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas de Información por los Usuarios*. Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010). Paper 89. Lima, Peru.

Mahemoff, M. (2006) *Ajax Design Patterns*. U.S.A.: O'Reilly

Marcos, E. y Marcos, A. (1998), *An Aristotelian Approach to the Methodological Research: a Method for Data Models Construction. En Information Systems-The Next Generation*. pp. 532-543. USA: L. Brooks & C. Kimble (Eds.). Mc Graw-Hill.

Martínez-Ruiz, F., Vanderdonckt, J. y Arteaga, J. (2010). *TRIAD: Triad-based Rich Internet Application Design*. First Int. Workshop on User Interface Extensible Markup Language UsiXML 2010. Berlin, Alemania.

Meixner, G., Paternò, F. y Vanderdonckt. J. (2011). *Past, Present, and Future of Model-Based User Interface Development*. i-com. 10(3), pp. 2-11. [Artículo en línea]. Disponible en: http://www.oldenbourg-link.com/doi/abs/10.1524/icom.2011.0026. [Consulta: 2012, junio 02]

Montilva, J., Barrios, J. y Rivero, M. (2008). *Gray Watch Método de Desarrollo de Software para Aplicaciones Empresariales*. Proyecto Methodius. Fonacit 2005000165. Mérida.

Moraga, M., Calero, C., Piattini, M. y Diaz, O. (2007). *Improving a portlet usability model*. Software Quality Control. 15(2), pp. 155-177.

Moreno, J., Bolaños, L. y Navia, M.(2010). Exploración de Modelos y Estándares de Calidad Para el Producto de Software. *UIS Ingenierías Revista de la Facultad de Ingenierías Fisicomecánicas*, 9(1), pp. 39 – 53.

Nielsen, J. y Molich, R. (1990). *Heuristic evaluation of user interfaces*. Proc. ACM CHI'90 Conference, pp. 249-256.

Nielsen, J. (1993). *Usability Engineering.* London. Academic Press

Nielsen, J. y Mack, R. (1994). *Usability Inspection Methods*, New York:U.S.A. John Wiley & Sons.

Nielsen, J (2005). *Ten Usability Heuristics*. [Página web en línea]. Disponible: http://www.useit.com/papers/heuristic/heuristic_list.html

Olsina, L. y Rossi, G. (2002). *Measuring Web Application Quality with WebQEM*. IEEE Multimedia. 9(4), pp. 20-29.

Ogata, S. y Matsuura, S. (2010). *Evaluation of a use-case-driven requirements analysis tool employing web UI prototype generation*, WSEAS Transactions on Information Science and Applications, 7(2), pp. 273-282.

O'Reilly, T., y Battelle, J. (2009). *Web Squared: Web 2.0 Five Years On*. Special Report for the Web 2.0. [Documento en línea]. Disponible en: http://assets.en.oreilly.com/1/event/28/web2009_websquared-whitepaper.pdf [Consulta: 2012, mayo 20].

Pfleeger, S. (2008). Software *Engineering: Theory and Practice*. (3a. ed.). Singapore: Pearson-Education.

Pressman, R. (2010). *Software Engineering: A Practitioner's Approach Pressman* (7a. ed.). New York, U.S.A.: McGraw-Hill.

Ralyté, J,, Deneckère, R. y Rolland, C. (2003). *Towards a generic model for situational method engineering*. 15th International Conference. CAiSE 2003. Austria, pp. 95-110.

Reijnders, G., Khadka, R.y Jansen, S. (2011). *Developing a legacy to SOA migration method*," Department of Information and Computing Sciences, Utrecht University, Reporte Técnico. UU-CS-2011-008.

Rodrigues, A. (2001). *Project Goals, Business Performance, and Risk*. U.S.A.: Cutter Consortium e-Project Management Advisory Service Executive.

Rooms, Christoph (2007) *Improving User Experience with Rich Internet Applications.* Software Engineering Today. Conference and Exhibition. Zurich.

Rossi, G., Fraternali, P. y Sánchez-Figueroa, F. (2010) *Rich Internet Applications. Internet Computing*. IEEE, 14(3), pp. 9-12. [Artículo en línea]. Disponible en: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5481362&isnumber=5481356. [Consulta: 2012, mayo 20].

Saeki, M. (2003). *Embedding metrics into information systems development methods: An application of method engineering technique*. CAiSE 2003. Austria, pp. 374-389.

Santos, F. (2010). *Sistema de Controlo de Versões em Mundos Virtuais para Negociação de Configurações Espaciais*. Tesis Doctoral. Universidad de Trás-os-Montes e Alto Douro. Vila Real, Portugal.

Scott, B. y Neil, T. (2009). *Designing Web Interfaces: Principles and Patterns for Rich Interactions*. U.S.A :O'Reilly

Shneiderman, B., y Plaisant, C. (2010). *Designing the user interface: Strategies for effective human-computer interaction* (5a. ed.). Boston, Mass: Addison-Wesley.

Sommerville, I. (2005). *Ingeniería del Software* (7a. ed.). Madrid, España: Pearson Educación.

The Standish Group (2009). *The CHAOS Report*. [Documento en línea]. Disponible: http://www.standishgroup.com/newsroom/chaos_2009.php [Consulta: 2012, marzo 15].

Universidad Pedagógica Experimental Libertador (2010). *Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales* (4a ed.). Caracas, Venezuela: Fondo Editorial de la Universidad Pedagógica Experimental Libertador.

Valdez, J. (2011). *La Minería de Datos en Educación Matemática. Relación entre Estilos de Aprendizaje y Desempeño académico*. Tesis de Maestría. Universidad Nacional de Colombia. Palmira, Colombia

Valverde, F. y Pastor, O. (2009). *Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach*. Web Information System Engineering. WISE 2009. Springer. 5802, pp. 131—144.

Vlaanderen, K., Valverde, F. y Pastor, O (2008). *Improvement of a web engineering*

method applying situational method engineering. 10th International Conference on Enterprise Information Systems. 1, pp. 147–154. Barcelona: España.

Weerd, I. Van de, Brinkkemper, S., Souer, J. y Versendaal, J. (2006). *A situational implementation method for web-based content management system-applications: method engineering and validation in practice.* Software Process: Improvement and Practice (edición especial), pp. 521-538.

Weerd, I. Van de, Weerd, S. de, y Brinkkemper, S. (2007). Situational Method Engineering: Fundamentals and Experiences. Capítulo: Developing a Reference Method for Game Production by Method Comparison, pp 313–327. Springer Boston.

Weerd, I. Van de, y Brinkkemper, S. (2009). *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. Capitulo III: Meta-modeling for situational analysis and design methods, pp. 38–58. Idea Global Publishing.

Westfall, L. (2011). Las Fallas en la Ingeniería de Requisitos. *Revista Facultad de Ingeniería USBMed*. 2(2). Universidad de San Buenaventura. Colombia.

ANEXO A

Currículum Vitae del Autor

Gerana Auxiliadora Espinoza Contreras

Nace en San Felipe, Edo. Yaracuy. Realizó estudios de Educación primaria en el Colegio "Andres Bello". Cursó sus estudios de Educación Secundaria y Diversificada en el Colegio "Fray Luis Amigó" de San Felipe – Edo. Yaracuy, donde obtiene el título de Bachiller en Ciencias. Posteriormente inicia su carrera universitaria en la Universidad Centroccidental "Lisandro Alvarado" allí obtiene el título de "Ingeniero en Informática" en el año 1999, obteniendo la mención Cum Laude. Ha realizado los siguientes cursos: Operador de Micro en la Fundación Educación Continua, Extensión Universidad Simón Bolívar (1994), Técnico Especialista en Redes en Mercadística Data Center (1999), Ingles Básico en FUNDAUC (2002-2003), Desarrollo de Aplicaciones usando Metodología y Herramientas Ágiles (2011) y el Proceso de la Investigación (2012) en la Universidad Centroccidental "Lisandro Alvarado". Los Diplomados de Entornos Virtuales de Aprendizaje (2011) y Docencia Universitaria (2011-2012) en la Universidad Centroccidental "Lisandro Alvarado" Posee experiencia profesional como preparador del Centro de Computación del Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado" desde el año 1996 a 1999, como Desarrollador de Sistemas, (pasantías) en la Universidad Nacional Experimental Politécnica "Antonio José de Sucre", UNEXPO, en el periodo marzo – mayo 1999, seguidamente ocupa el cargo de Coordinadora del Departamento de Desarrollo de Software en la empresa Consultores Asociados en el lapso 1999-2003, posteriormente ejerce como Desarrollador de Aplicaciones Web y Sistemas de Información desde 2004 al 2006, en TURBOSOFT Aplicaciones Informáticas en Barcelona-España. También se desempeñó como Analista Desarrollador de Sistemas en la empresa CIDE, S.A. en el período 2006-2009, seguidamente ocupa el cargo de Analista Desarrollador de Aplicaciones Web en el Instituto de Desarrollo Rural (INDER) en los años 2009-2010. Actualmente trabaja como Profesor Ordinario a Tiempo Completo en el Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", desempeñándose como Docente en el área de Laboratorios de la carrera Ingeniería en Informática.