

El Catálogo de Requisitos en la Ingeniería de Requerimientos

PERCY OSCAR HUERTAS NIQUÉN

YASIEL PÉREZ VERA

GIOVANNI ROLANDO CABRERA MÁLAGA



PRIMERA EDICIÓN



**EDITORIAL
UNSA**

El Catálogo de Requisitos en la Ingeniería de Requerimientos



**EDITORIAL
UNSA**

El Catálogo de Requisitos en la Ingeniería de Requerimientos

Percy Oscar Huertas Niquén
Yasiel Pérez Vera
Giovanni Rolando Cabrera Málaga

Primera Edición



**EDITORIAL
UNSA**

El Catálogo de Requisitos en la Ingeniería de Requerimientos

Elaboración del contenido

© Percy Oscar Huertas Niquén

Yasiel Pérez Vera

Giovanni Rolando Cabrera Málaga

Editado e Impreso en:

Universidad Nacional de San Agustín de Arequipa

Arequipa-Perú

Editorial UNSA

Calle Paucarpata, Puerta 2, Área de ingenierías

Teléfono: 959637044

E-mail: editorial@unsa.edu.pe

Primera edición, junio 2024

Tiraje: 500 ejemplares

Hecho en Depósito Legal en la Biblioteca Nacional del Perú N°: 2024-06019

ISBN: 978-612-5136-24-4

Reservados todos los derechos. No se permite la reproducción total o parcial de esta obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de los titulares del copyright. La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.

IMPRESO EN PERÚ

Agradecimiento

La elaboración de un texto académico no es una tarea fácil, implica un enorme sacrificio que involucra a muchas personas. Un especial agradecimiento a mi esposa Jackelinne López Torres por su paciencia mientras realizaba esta ardua tarea.

M. Sc. Percy Oscar Huertas Niquén

Agradezco al M. Sc. Percy Huertas Niquén por invitarme a formar parte de este interesante proyecto que consiste en la confección del presente texto académico.

Dr. Yasiel Pérez Vera

Agradezco a Dios por todas las bendiciones que llegan a mi vida, al M. Sc. Percy Huertas Niquén por su enseñanza, guía e invaluable gesto al permitirme participar en la preparación de este texto académico, a mis amados hijos Giovanni, Renato y Nathaly por llenar de alegría mis días y, a mi esposa Milané, por su paciencia y compañía.

M. Sc. Giovanni Rolando Cabrera Málaga

INDICE

Agradecimiento	IX
Introducción.....	XV
Sobre los autores.....	XVII
CAPÍTULO 1	1
Ingeniería de Requisitos	1
1.1 Introducción	1
1.2 Definición.....	2
1.3 El problema de los requisitos.....	4
1.4 Requerimientos vs requisitos	9
1.5 Clasificación de los requisitos	12
1.6 El problema de la inconsistencia y ambigüedad	15
1.7 Gestión del cambio en los requisitos	17
1.8 Herramientas de soporte	20
CAPÍTULO 2.....	33
Elicitación de Requisitos.....	33
2.1 Introducción	33
2.2 Definición.....	33
2.3 Objetivo de la elicitation	34
2.4 Problemas de la elicitation	34
2.5 Funciones de la elicitation	37
2.6 Técnicas de la elicitation	38
2.7 Proceso de la elicitation	41
2.8 Metodologías para eliciar requisitos.....	42
2.9 Relación con los estándares	44
CAPÍTULO 3.....	47
El Catálogo de Requisitos	47
3.1 Introducción	47
3.2 Definición.....	48
3.3 Estructura del catálogo de requisitos	48
3.4 Artefactos de apoyo al catálogo de requisitos	51
3.5 Plantillas de trabajo.....	52
3.0.1 Organizaciones.....	52
3.0.2 Actores.....	53
3.0.3 Autores	54
3.0.4 Fuentes	55
3.0.5 Educación.....	56
3.0.6 Ilación	57
3.0.7 Especificación.....	59
3.0.8 Trazabilidad	60
3.6 Procedimiento para elaborar el catálogo de requisitos	62
3.7 Ejemplo de catálogo de requisitos.....	62
CAPÍTULO 4	85
Educación de Requisitos	85
4.1 Introducción	85
4.2 ¿Qué es la educación de requisitos?	85

4.3	Proceso de educación de requisitos.....	86
4.4	Técnicas para la educación de requisitos	87
4.5	Desventajas de la educación de requisitos	89
4.6	Ejemplo de educación de requisitos.....	89
CAPÍTULO 5.....		97
Ilación de Requisitos		97
5.1	Introducción	97
5.2	¿Qué es la ilación de requisitos?	97
5.3	Proceso de ilación de requisitos.....	98
5.4	Técnicas para la ilación de requisitos.....	98
5.5	Desventajas de la ilación de requisitos.....	99
5.6	Ejemplo de ilación de requisitos	99
CAPÍTULO 6.....		105
Especificación de Requisitos.....		105
6.1	Introducción	105
6.2	¿Qué es la especificación de requisitos?.....	105
6.3	Proceso de especificación de requisitos.....	106
6.4	Técnicas para la especificación de requisitos	106
6.5	Desventajas de la especificación de requisitos	107
6.6	Ejemplo de especificación de requisitos.....	108
CAPÍTULO 7.....		115
Trazabilidad de Requisitos		115
7.1	Introducción	115
7.2	Elementos de la trazabilidad.....	116
7.3	Proceso de trazabilidad.....	117
7.4	Trazabilidad simple	118
7.5	Trazabilidad múltiple	119
7.6	Implementación de la trazabilidad.....	120
7.7	Ánálisis de la trazabilidad.....	121
7.8	Ejemplo de trazabilidad.....	121
CAPÍTULO 8.....		127
Requisitos no Funcionales		127
8.1	Introducción	127
8.2	Definición.....	130
8.3	Clasificación	130
8.4	Efecto de los requisitos no funcionales.....	133
8.5	Ejemplo de requisitos no funcionales.....	133
CAPÍTULO 9.....		137
Gestión del Riesgo en los Requisitos		137
9.1	Introducción	137
9.2	Definición.....	138
9.3	Riesgos en el desarrollo de software	139
9.4	Descripción del proceso.....	142
9.5	Diseño de la plantilla de trabajo	144
9.6	Ejemplo de gestión del riesgo	146
CAPÍTULO 10.....		149
Requisitos y Pruebas de Software		149

10.1	Introducción	149
10.2	Pruebas de software y el catálogo de requisitos.....	152
10.3	Ejemplo de la definición de pruebas de software	153
CAPÍTULO 11.....		157
Métricas y Requisitos		157
11.1	Introducción	157
11.2	Definición.....	158
11.3	Proceso de elaboración de métricas	158
11.4	Métricas sugeridas	159
11.5	Ejemplo de cálculo de métricas	160
CAPÍTULO 12		165
Inconsistencias y Ambigüedades.....		165
12.1	Introducción	165
12.2	Definiciones.....	167
12.3	Proceso de corrección	167
12.4	Procedimiento para corregir en la educación.....	168
12.5	Procedimiento para corregir en la ilación	170
12.6	Procedimiento para corregir en la especificación	171
12.7	Ambigüedades comunes	171
12.0.1	Surgidas por la descomposición de la aplicación	172
12.0.2	Surgidas por abuso de la abstracción	174
12.0.3	Surgidas por el uso de interfaces gráficas de usuario.....	175
12.0.4	Surgidas por la reutilización de plantillas.....	176
12.0.5	Surgidas por el mal diseño de los casos de uso	177
12.0.6	Surgidas por la tecnología de objetos.....	178
12.0.7	Surgidas por las iteraciones	179
12.0.8	Surgidas por la arquitectura	181
12.0.9	Surgida por el uso de escenarios.....	181
12.0.10	Mala interpretación de los dominios del sistema.....	182
12.0.11	Surgidas por la educación de requisitos	184
12.0.12	Surgidas por la ilación de requisitos.....	185
12.0.13	Surgidas por la especificación de requisitos	187
12.8	Análisis de inconsistencias y ambigüedades	188
12.9	Solución a las inconsistencias y ambigüedades	189
12.10	Confección del catálogo de requisitos	190
12.11	Ejemplo del proceso de corrección	191
12.0.14	Ejemplo de inconsistencia	191
12.0.15	Ejemplo de ambigüedad.....	192
Bibliografía.....		195

Introducción

El primer semestre del año académico 2017, el Departamento Académico de Ingeniería de Sistemas e Informática de la Facultad de Ingeniería de Producción y Servicios de la Universidad Nacional de San Agustín propone dictar el curso de Ingeniería de Requerimientos bajo el principio de otorgar conocimiento para una correcta construcción de software basado en criterios de calidad a partir de la fase de análisis, es decir en etapas tempranas de la construcción de un producto de software.

La experiencia adquirida en las empresas Objetive Software y Software Solutions S. A. C. permitieron establecer un ambicioso plan para entregar conocimientos sobre la elaboración de catálogos de requisitos. Si la estrategia fallaba entonces existía dificultad en lograr que los estudiantes alcanzaran la concepción de construir software con el principio de calidad, perdiendo el sentido de los criterios arquitecturales del software, caso contrario tendrían la posibilidad de construir, paso a paso, un catálogo de requisitos.

El punto de partida es establecer pruebas de entrada para lograr determinar el estado situacional sobre la confección de los catálogos de requisitos, para que, a partir de ello, se estudiase la forma de cómo enfrentar a la gran problemática de organizar información. No se desea que el estudiante se enfrente a engorrosas preguntas teóricas, por el contrario, se opta por la confección de un artefacto que cumpliera este único propósito.

Como parte del procedimiento, se solicita a los estudiantes resolver una ecuación cuadrática (fig. 1) conceptualizando un pequeño programa bajo el lenguaje de programación que dominan o el empleo de cualquier otra herramienta que pudiere resolver lo solicitado.

$$-x^2 + 7x - 10 = 0$$

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figura 1: Ecuación cuadrática a resolver

Con el entusiasmo caracterizado, los estudiantes codifican el problema entregando las raíces rápidamente, pero llama poderosamente la atención que no hicieran pregunta alguna con respecto al problema a resolver. Seguidamente los estudiantes exponen sus resultados entregando respuestas numéricas correctas. Normalmente, el 100 % de ellos no hacen pregunta alguna con respecto a la necesidad de presentación de la solución.

Seguidamente se muestra lo que se necesita como respuesta (fig. 2) quedando impresionados porque a pesar de que la solución era la misma, la presentación se mostraba diferente. En este punto se hace la pregunta: ¿Quién cometió el error?, ¿El usuario? o ¿El analista? Entonces nos encontramos frente a un grave problema; no logran comprender el objetivo del catálogo de requisitos ya que mantienen el análisis clásico: Escuchar y hacer, sin consultar las necesidades del usuario final.

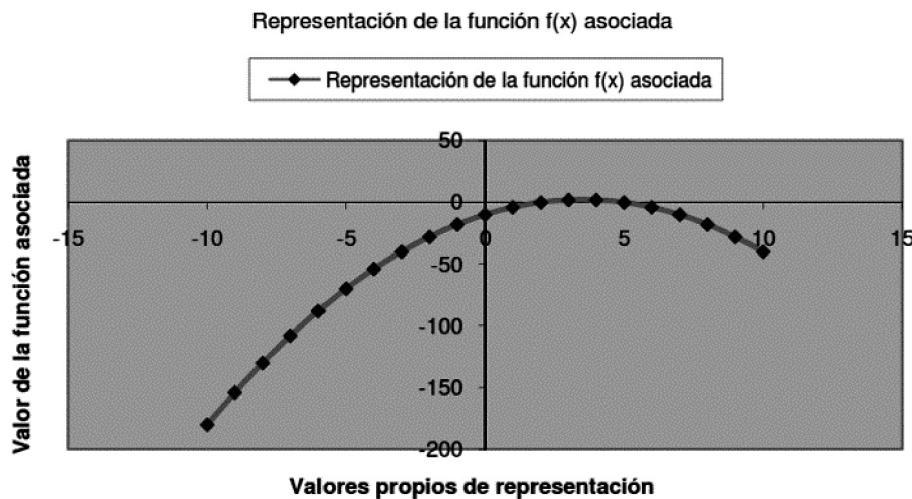


Figura 2: Resultado de la ecuación cuadrática solicita

Este mismo esquema se repite en el curso de Diseño y Arquitectura de Software al igual que en Pruebas de Software. Todo lo hacen sin consultar las necesidades del usuario. Los diagramas de UML o artefactos de análisis o diseño los obtienen después de la codificación. A partir de esta experiencia queda la tarea de planificar un curso donde conocieran la real importancia de los requisitos y entendieran que desde este artefacto se pueden generar criterios de calidad e inclusive disminuir los costos en la construcción de productos de software.

Para alcanzar el objetivo, en cada semestre se consignan proyectos de desarrollo de software reales. Cada grupo de trabajo lleva a cabo actividades de gestión para planificar sus proyectos y entregar los artefactos en hitos definidos. Esta experiencia impulsa el deseo de escribir el presente texto, que consideramos un documento que tomarán en cuenta los estudiantes, así como desarrolladores de productos de software. Aunque no se pretende una visión teórica exhaustiva ni su relación con técnicas o métodos existentes, el único objetivo es que el estudiante adquiera los conocimientos sobre "CÓMO ELABORAR UN CATÁLOGO DE REQUISITOS".

Sobre los autores

M.SC. PERCY OSCAR HUERTAS NIQUÉN



Licenciado en Estadística por la Universidad San Martín de Porres y Magíster en Ingeniería de Sistemas con mención en Ingeniería de Software por la Universidad Nacional de San Agustín. Realizó estudios de maestría en Ciencia de la Computación en la Universidad de Chile y una segunda especialidad en Ingeniería de Sistemas en la Universidad Nacional de San Agustín; así como el Doctorado en Ingeniería de Software en Atlantic International University y el Doctorado en Estadística e Informática en la Universidad Nacional del Altiplano; actualmente es docente universitario en la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín y ex-Decano de la Facultad de Ingeniería en la Universidad La Salle de Arequipa. Gerente General de Software Solutions SAC y consultor para diversas empresas de construcción de software. Asumió el cargo de Director de la Oficina Universitaria de Informática y Sistemas de la Universidad Nacional de San Agustín y actualmente es Director de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín que bajo su dirección se obtuvo la acreditación ABET.

DR. YASIEL PÉREZ VERA



Doctor en Business Administration (DBA) en la Universidad Nacional de San Agustín de Arequipa. Obtuvo su grado de Magíster en Gestión de Proyectos Informáticos en la Universidad de las Ciencias Informáticas en La Habana, Cuba. Titulado en la misma institución en el año 2014 como Ingeniero en Ciencias Informáticas con la condición de título de oro. Actualmente es profesor en la Carrera Profesional de Ingeniería de Software de la Facultad de Ingeniería de la Universidad La Salle de Arequipa. Ocupa el cargo de Director de la Escuela de Posgrado de la Universidad La Salle. Se desarrolla como profesor de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa. Es editor jefe de la Revista Científica Innovación y Software de la Universidad La Salle. Sus intereses de investigación son la gestión de proyectos, ingeniería de software,



inteligencia artificial, redes neuronales y minería de datos.

M.SC. GIOVANNI ROLANDO CABRERA MÁLAGA

Maestro en Ciencias: Informática con mención en Tecnologías de la Información y la Comunicación para la Gestión y la Educación por la Universidad Nacional de San Agustín de Arequipa, Ingeniero de Sistemas por la Universidad Católica de Santa María. Miembro Ordinario del Colegio de Ingenieros del Perú. Cuenta con Certificaciones Internacionales Microsoft: MCP (Microsoft Certified Professional), MCTS (Microsoft Certified Technology Specialist) y MCPD (Microsoft Certified Professional Developer). Asesor de Desarrollo de Software y Administrador de Bases de Datos (DBA) para instituciones públicas y privadas. Actualmente desempeña sus labores como Docente Auxiliar Nombrado en el Departamento Académico de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa.

CAPÍTULO 1

Ingeniería de Requisitos

1.1 Introducción

Hoy en día, las empresas desarrolladoras de software están empezando a preocuparse por administrar las necesidades del cliente, en cuanto a software, permitiendo construir productos de software con criterios de calidad. Aunque tienen presente el problema, aún encuentran que se necesita una mayor explicación del tema; con mayor preocupación notan el problema del efecto de la mala distribución de las necesidades del cliente expresadas en información.

El estudio realizado por el Standish Group [1] en el año 2002 concluye que el 15 % de los proyectos de software fracasan en su totalidad, el 51 % fueron considerados como "concluidos tarde". Solo el 54 % de las características definidas originalmente de un proyecto fueron entregadas, y de ellas el 45 % nunca se emplearon. Los investigadores estiman que el problema se encuentra en la mala definición de los requisitos.

Las razones fundamentales de las fallas en los proyectos de software son atribuibles a la planificación y seguimiento del proyecto en sí mismo. Algunos desarrolladores solo brindan un servicio directo a la calidad, el trabajo en equipo y la mejora continua y no están dispuestos a invertir en capacitación y práctica para crear y mantener una "*cultura de mejora de la calidad*" en beneficio de proyectos futuros.

En el año 2015 el Standish Group reporta que la tasa de fracaso de los proyectos relacionados a tecnología representa más del 70 %. Los resultados indican que el 31 % de los proyectos son anulados antes de ser culminados. El 52.7 % de los proyectos incrementan sus costos en un 189 % a partir de sus presupuestos originales y el 50 % de estos son puestos en producción y entran en operación.

Hace hincapié que el proyecto de desarrollo promedio incrementa su fase de programación o planificación en un 50 % o más y que el 75 % de los productos de software, considerados

grandes, fueron concluidos sin cumplir con las necesidades del cliente o con una mala especificación de los requisitos funcionales (no fueron consideradas las inconsistencias ni las ambigüedades) y sin tomar en cuenta los requisitos no funcionales.

Las fallas más notorias son: Requisitos incompletos (13.1 %), deficiencias con el involucramiento del usuario (12.4 %), deficiencias en los recursos (10.6 %), deficiencias en el soporte ejecutivo (9.3 %), cambios en los requisitos y sus especificaciones (8.7 %), deficiencias en la planeación (8.1 %), el proyecto no necesita más cosas (7.5 %), deficiencias en la administración de Tecnologías de Información (6.2 %), desconocimiento de la tecnología (4.3 %) y otros (9.9 %).

En esencia, una mala especificación de los requisitos permite que el proyecto de software pueda asumir grandes errores dando como consecuencia una baja calidad en su construcción y en el producto. Los usuarios sentirán inconformidad siendo, bastante probable, que el producto de software construido jamás sea empleado y que a pesar de poseer los mejores elementos en su construcción este sea tildado de *"malo"*.

De esta manera, podemos indicar que la ingeniería de requisitos personaliza un aspecto fundamental en la construcción de un producto de calidad y su principal actividad consiste en producir requisitos consistentes, y sin ambigüedades, que sirvan de apoyo para comprender el problema; asimismo disminuir los riesgos y reducir costos en el desarrollo del producto de software permitiendo insertar criterios de calidad [2].

Es muy importante entender y conocer el dominio del problema y la organización actual desde el ángulo de vista del flujo de información y sus transacciones antes de reconocer y establecer los requisitos. Desarrollar un sistema sin entender y comprender sus principales características puede hacer que el producto construido no sea de la satisfacción del cliente, ni de los usuarios.

I. Dávila [2] asevera que *"para obtener requisitos correctos y sin ambigüedad existen diversos estándares y normas. Los más reconocidos son: Especificaciones de requisitos del Software (IEEE, 2000), Métrica versión 3 (Ministerio de Política Territorial y Administración Pública, s.f.) en España, Requisitos de Software (IEEE, 2004), Estándar CMMI (Software Engineering Institute, 2006), Gestión de Sistemas de Calidad (ISO, 2008). Estos estándares y patrones ayudan a mejorar la calidad de los requisitos de desarrollo de software locales"*. Ante lo mencionado, se puede indicar que estos estándares no presentan un procedimiento concreto de como eliminar la complejidad del modelo de negocio y presentar información relevante y sostenible de los requisitos.

1.2 Definición

Según la Real Academia de la Lengua Española (RAE) [3], la Ingeniería se define como el *"Conjunto de conocimientos orientados a la invención y utilización de técnicas para el*

aprovechamiento de los recursos naturales o para la actividad industrial". Asimismo, la RAE [3] define la palabra requisito como "Circunstancia o condición necesaria para algo".

De acuerdo con el entendimiento de ambas definiciones, se puede deducir que la Ingeniería de Requisitos es "el conjunto de procesos que permiten recopilar, organizar, analizar, verificar y validar las necesidades del cliente con la finalidad de construir un producto de software con características de calidad".

El Instituto de Ingeniería Eléctrica y Electrónica (IEEE) [4] define a la Ingeniería de requisitos como "la capacidad que posee el usuario para resolver un problema, así como la capacidad que debe poseer un sistema para satisfacer características de calidad en base a una documentación sólida".

P. Zave [5] la define como "La Ingeniería de Requisitos es la rama de la Ingeniería de Software que tiene que ver con las metas del mundo real para proveer servicios y restricciones en un grande y complejo sistema de software. También concierne a las relaciones entre los factores para la especificación precisa del comportamiento del sistema, y para su evolución de familias de sistemas.".

P. Loucopoulos y V. Karakostas [6] proponen dos definiciones. La primera donde menciona que "La Ingeniería de Requisitos se encarga de todas las actividades que intentan comprender las necesidades exactas de los usuarios de un sistema software y traducir tales necesidades a especificaciones precisas y no ambiguas para que posteriormente puedan ser usadas en el desarrollo de un sistema.". La segunda, donde indica que "La Ingeniería de Requisitos puede ser definida como el desarrollo sistemático de los requisitos a través de un proceso iterativo y cooperativo en el que se analiza el problema, se documenta el resultado en diversos formatos de representación, y se comprueba la exactitud de la comprensión alcanzada.".

R. Thayer, S. Bailin, & M. Dorfman [7] indican que "La Ingeniería de Requisitos facilita los mecanismos adecuados 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".

O. Hurtado [8] define a la Ingeniería de Requisitos como "la rama de la Ingeniería de Software que se encarga de la captación, análisis y verificación de la información relativa a las necesidades de los usuarios; de la invención y especificación correcta y completa de los requisitos propiamente dichos dentro de un proceso evolutivo y de negociación; y del mantenimiento o gestión de los requisitos dentro de todo el proceso de desarrollo de software. El producto o activo del proceso de la Ingeniería de Requisitos es el documento de especificación de requisitos".

I. Sommerville [9] también propone dos definiciones. En la primera indica que es un "Proceso que comprende todas las actividades de requerimientos para crear y mantener un documento de requerimientos del sistema"; y en la segunda menciona que es un "Proceso de aplicar un

método estructurado, el cual analiza el sistema y desarrolla un conjunto de modelos gráficos del mismo que actúan como una especificación del sistema”.

R. Pressman [10] menciona que “La ingeniería de requerimientos proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional. Incluye siete tareas diferentes: concepción, indagación, elaboración, negociación, especificación, validación y administración. Es importante notar que algunas de estas tareas ocurren en paralelo y que todas se adaptan a las necesidades del proyecto.”.

K. Pohl and C. Rupp [11] mencionan que “la Ingeniería de Requisitos es un enfoque sistemático y disciplinado para la especificación y gestión de requisitos con el objetivo de conocer los requisitos relevantes, lograr un consenso entre las partes interesadas sobre estos requisitos, documentarlos de acuerdo con las normas dadas y gestionarlos sistemáticamente; también de comprender y documentar los deseos y necesidades de los interesados, especificando y gestionando los requisitos para minimizar el riesgo de entregar un sistema que no cumpla con los deseos y necesidades de los interesados”.

El Swebok [12] hace una definición de requisitos de software como “En su forma más básica, un requisito de software es una propiedad que debe ser exhibida por algo para resolver algún problema del mundo real. Puede apuntar a automatizar parte de una tarea para que alguien apoye los procesos comerciales de una organización, para corregir las deficiencias del software existente o para controlar un dispositivo, por nombrar solo algunos de los muchos problemas para los que son posibles soluciones de software.”.

R. Young [13] menciona que “Un requisito es un atributo necesario en un sistema, una declaración que identifica una capacidad, característica o factor de calidad de un sistema para que tenga valor y utilidad para un cliente o usuario. Los requisitos son importantes porque proporcionan la base para todo el trabajo de desarrollo que sigue. Una vez que se establecen los requisitos, los desarrolladores inician el otro trabajo técnico: diseño, desarrollo, prueba, implementación y operación del sistema.”.

Teniendo en cuenta las definiciones antes mencionadas, se concluye que el objetivo de la Ingeniería de Requisitos es la especificación, a través de un proceso sistemático y evolutivo, de lo que debe hacer un producto de software y de las restricciones que condicionan como ha de ser implementado para garantizar que el producto final esté acorde con las necesidades del cliente cumpliendo, de esta manera, con un conjunto de criterios de calidad.

1.3 El problema de los requisitos

Pocos textos presentan un enfoque real a la problemática de los requisitos, la gran mayoría de ellos se dedican a explicar sus ventajas y la forma como gestionarlos y administrarlos.

Algunos textos presentan alguna problemática soslayando sus verdaderos problemas y el crucial efecto en la construcción de productos de software y en donde quedan perjudicados los criterios de calidad; muchos otros se han dedicado a la conceptualización de métodos que permitan una eficaz gestión de los requisitos.

Cerca de 40 años de trabajo, en la construcción de software, han permitido el entendimiento de la problemática de los requisitos, sus ventajas y desventajas y como la complejidad del problema a resolver influye en su concepción y en las metodologías y modelos que proponen los textos. Se han cuestionado artefactos expuestos en textos, llegando a expresar: ¿Para qué sirve esto?; posteriormente, y después de un largo periodo, se logró entender la verdadera problemática que surge producto de la situación en que se vive. Los problemas se resumen a continuación.

1. El temor de los que recién comienzan: Por ahorrar costos, los jefes de proyectos envían a los analistas junior para realizar las entrevistas. Su inexperiencia hace que cometan errores que son introducidos en las primeras concepciones de los requisitos. Luego cuando los analistas se dan cuenta de los errores culpan a los iniciados, pero no se dan el trabajo de planificar las entrevistas con los clientes ni proporcionar las estrategias para culminar con un correcto catálogo.
2. La falta de percepción del problema por parte de los experimentados: Comúnmente, las personas experimentadas cometen el error de enviar a los iniciados para posteriormente tomar lectura de las notas de los iniciados y proceder a corregirlos. Esto permite que el entendimiento del modelo del negocio quede lejos de su entendimiento real permitiendo que se introduzcan serios errores en su concepción.
3. La falta de dimensionamiento de la complejidad del problema a resolver: Es común pensar que todos los productos de software son construidos bajo un mismo estilo. La complejidad del problema a resolver es un factor gravitacional en la construcción del producto ya que se debe tener una visión estricta del modelo del negocio y de los artefactos a emplear para su solución. La visión que se obtuvo de un problema simple es común que sea introducido en problemas complejos.
4. La falta de uso de técnicas para elicitar requisitos: Muchos analistas no planifican las entrevistas, el diseño de cuestionarios u otras actividades dirigidas a obtener información para elaborar requisitos. Por otro lado, los analistas se acostumbran a "escuchar y hacer" pensando que de esta manera pueden obtener requisitos consistentes y sin darse cuenta de que introducen severas inconsistencias. La importancia de contar con un conjunto de herramientas es grande ya que permite ordenarnos en el proceso de construcción.
5. Falta de comunicación entre los integrantes del grupo de analistas: Es común encontrar como integrantes de este grupo a personas introvertidas, que sienten temor a conversar con sus compañeros y que presentan condiciones limitadas a pesar de los grandes conocimientos que poseen. Este defecto no permite una adecuada transmisión de información; la información es tan pobre que es fácil introducir inconsistencias o ambigüedades en los requisitos.

6. Inadecuado liderazgo del jefe de analistas: En algunas ocasiones, el jefe de analistas piensa que sus ideas son las mejores e imponen sus criterios sin dejar opinar a los demás integrantes. Esta imposición afecta severamente la forma como se pueden obtener los requisitos y puede introducir ambigüedades e inconsistencias en los mismos; es probable que algunos de los integrantes puedan tener mejores criterios o derroteros como visión de construcción.
7. Imposición de lo aprendido: En otras oportunidades los integrantes han aprendido en aulas universitarias artefactos que son dominados en su integridad e intentan usarlos en problemas reales sin saber si su uso proporciona resultados adecuados. Este problema se debe a que no analizan la complejidad del problema a resolver y piensan que lo aprendido se aplica a todo tipo de construcción de software. Este defecto produce enormes efectos negativos en la elaboración del catálogo de requisitos.
8. Todos los problemas se resuelven de la misma manera: En muchas ocasiones los encargados del grupo de analistas usan las mismas técnicas en todos los proyectos de desarrollo de software. Si no reconocen el tipo de problema al que se enfrentan no podrán determinar el artefacto que van a emplear; al emplear el mismo artefacto se tiene una alta probabilidad de introducir inconsistencias.
9. Los problemas personales: Cuando alguno de los integrantes del equipo de analistas presenta problemas personales es común que se sienta retraiðo, hecho que impide elicitar buenos requisitos. La transmisión de información se vuelve vaga debido a la cantidad de errores que se introducen durante la actividad y el entendimiento de la información se torna demasiado pobre. El líder del proyecto siempre debe estar atento a estos problemas.
10. Problemas producidos por la abstracción: Los analistas comúnmente tienden a abstraer detalles desde el punto de partida de las entrevistas, y traen consigo un conjunto de ideas que piensan que son las soluciones al problema planteado. Generalmente no se espera el momento adecuado ya que la ansiedad embarga por culminar con éxito las tareas encomendadas para luego codificarlas. Este problema es bastante común en los iniciados.
11. Problemas introducidos por el mal uso de los Casos de Uso: La mala utilización de los casos de uso introduce deficiencias en la información. Se piensa que ellos son intocables y no se permite la modificación de la estructura logrando, de esta manera, introducir graves errores en la concepción del problema. Estos pueden ser alterados dependiendo de la complejidad del problema a resolver y de las estrategias que va a utilizar el líder del proyecto.
12. La mala conceptualización del problema: Una pésima conceptualización del problema inserta el defecto al suponer tareas, actividades y procesos que a la larga resultan no ser parte de la solución. Las ideas poco claras tienden a insertar severos errores en la elaboración de requisitos haciendo que se piense en un modelo del negocio correcto cuando realmente nos encontramos equivocados. Esta visión produce una mala concepción de los módulos que integran el sistema final.

13. Inadecuado uso de la tecnología: Generalmente se encuentran herramientas que son usadas para todos los proyectos de software, sin tomar en cuenta el modelo del negocio al que se enfrentan. Las herramientas deben ser analizadas para encontrar sus ventajas y desventajas y determinar sobre qué tipo de proyectos de software deben ser empleadas evitando el uso de estructuras diferentes de toma de información.
14. Las costumbres de los clientes: Las costumbres de los clientes insertan vicios en la forma como se logra tomar la información. Las particularidades que presentan los clientes como la forma de vestir, la forma de expresarse, sus invitaciones a alguna actividad muchas veces causan intimidación; factor que es considerado negativo a la hora de eliciar requisitos permitiendo la introducción de inconsistencias en los mismos.
15. Inadecuado uso de la ética profesional: Debido a factores externos o ajenos al equipo de analistas, estos soslayan las recomendaciones técnicas dadas por el jefe del proyecto entregando artefactos que supuestamente se encuentran bien elaborados cuando en el fondo resultan incompletos para la transmisión de información permitiendo la introducción de una serie de errores en los requisitos y una pérdida de tiempo en su revisión.
16. No escribir a tiempo la información obtenida: Varios analistas tienen la pésima costumbre de "*dejar para mañana lo que se puede hacer hoy*". Guardan la información pensando que en los siguientes días la misma se va a encontrar tan fresca como se obtuvo. Finalmente, cuando intentan redactarla no incluyen factores que en su primer momento estuvieron presentes en los relatos de los usuarios finales.
17. La mala redacción: Una mala redacción de los documentos que soportan los requisitos permite una mala interpretación soliendo ser fácil introducir inconsistencias y ambigüedades en los mismos. Estos errores suelen afectar a etapas posteriores de la construcción del catálogo de requisitos, inclusive la concepción de artefactos asociados, sufriendo severos daños en la trazabilidad simple o múltiple de los mismos requisitos.
18. Mal uso de las técnicas escogidas: Es bastante común que los integrantes del equipo de analistas suelan interpretar a su manera el uso de las técnicas asociadas a la elicitación de requisitos. Este comportamiento suele dejar efectos colaterales en todas las fases de construcción del catálogo de requisitos haciendo que la codificación se viere afectada por el mismo problema. La discusión de estos importantes temas debe ser socializada con el resto del equipo.
19. No tomar en cuenta a los verdaderos especialistas de la información: También es común dejar de lado a los verdaderos agentes que trasladan la información y que conocen la funcionalidad de los procesos. Aunque en sí no se encuentran embebidos de ellos, conocen detalles importantísimos del flujo de información y de los tiempos asumidos en la solución de actividades internas.
20. No tomar en cuenta los requisitos no funcionales: Este es otro problema común, la desesperación por comenzar el trabajo hace que los analistas se enfoquen en los requisitos funcionales, dejando de lado o para el final, los requisitos no funcionales. Los requisitos no funcionales aportan altos criterios de concepción sobre la forma como

- deben definir los requisitos funcionales y otorgan el sendero para la construcción de un correcto software.
21. Tiempos cortos para la elicitation de requisitos: Normalmente la presión ejercida sobre el jefe de proyecto para que la construcción del producto se haga en el más corto plazo permite que la elicitation de requisitos no se planifique con sumo cuidado. Al no ser así redactan requisitos vagos los mismos que son mal interpretados al momento de la concepción del producto permitiendo que el codificador inserte criterios personales que no se encontraban contemplados en el catálogo.
 22. Falta de costumbre por calcular métricas: No es muy común que al final del proyecto el jefe de este tome conciencia en el cálculo de métricas. Estas medidas casi siempre son dejadas de lado porque no saben cómo usarlas o interpretarlas para sus futuros proyectos. Esta actividad, post proyecto, debe ser planificada de tal manera que se obtengan medidas que puedan servir para futuros proyectos de construcción de software. Este es el objetivo principal de las métricas.
 23. No tomar en cuenta toda la documentación de la organización: Los analistas dejan de lado la documentación interna de la organización y se centran en planificar entrevistas y elaborar cuestionarios. Muchas veces los oficios, cartas, resoluciones, directivas, memorándums y otros documentos internos son mucho más importantes que una entrevista incluido el mismo cuestionario. Estos documentos suelen proporcionar más información que las propias entrevistas.
 24. El empleo de métodos ágiles: La premura por la entrega del producto hace que se decidan por la utilización de métodos ágiles, pero suele suceder que los analistas tengan poca experiencia sobre su filosofía adoptando criterios que deslindan con los de los métodos ágiles. Esto permite que los requisitos sean tratados en diferentes fases y de diferentes maneras, no existiendo una coherencia entre las fases de construcción del catálogo de requisitos.
 25. El efecto de los sentimientos: El enfado, el odio, la tristeza, la indignación, la envidia, la venganza y los celos, así como la euforia, la paciencia, la admiración, el afecto, el optimismo, la gratitud, la satisfacción, el amor y el agrado contribuyen en la elicitation de requisitos. Un analista triste tiende a decaerse por lo que es altamente probable la inserción de errores. Un analista demasiado efusivo puede pasar por alto detalles importantes para la elicitation.
 26. La falta de tiempo de los clientes: El aplazamiento de los días y horas de las entrevistas o aplicación de cuestionarios tienden a desgastar a los analistas. Al final se sienten tan desanimados que no toman mucha importancia a la información obtenida haciendo que se introduzcan errores en el catálogo de requisitos. Es común encontrar los desplazamientos de tiempo en intervalos frecuentes. Esto es común porque no se planifica la actividad.
 27. Inadecuada reutilización de requisitos: Existen oportunidades en las que el jefe de analistas trae consigo esquemas o artefactos empleados en proyectos de desarrollo anteriores, pensando en que su uso les permite un ahorro sustancial de tiempo. Esta

- adecuación puede insertar efectos inesperados con la consiguiente pérdida de tiempo y obteniendo un catálogo de requisitos plagado de errores.
28. Baja calidad de las entrevistas: Una mala planificación de las entrevistas puede hacer que el entrevistado se aburra producto de la falta de conceptualización o redundancia. También influye la cantidad de tiempo que se lleva a cabo en la entrevista. Si el tiempo empleado es alto y el entrevistado siente que nada se ha dicho entonces es altamente probable que en las siguientes oportunidades el entrevistado se niegue a entregar información o decida no atender a los analistas.
29. Mal diseño de cuestionarios: La mala calidad de las preguntas o la gran cantidad de ellas pueden hacer que el que responda el cuestionario se aburra y no entregue la información adecuada. Los cuestionarios deben ser elaborados siguiendo técnicas para su construcción de tal forma que, con pocas preguntas y bien redactadas, se pueda obtener la máxima información posible.
30. Inserción de nuevos requisitos: Cuando las cosas no son claras y la construcción del producto de software no presenta una adecuada planificación es altamente probable que los clientes inserten nuevas necesidades. Estas necesidades pueden generar efectos colaterales serios a un conjunto de requisitos ya tomados logrando una inestabilidad del catálogo de requisitos por su alta inserción de errores. Los analistas deben acostumbrarse a llevar a cabo una buena gestión del cambio.
31. La condescendencia del jefe de analistas ante la constante modificación de requisitos: Ante la constante inserción de requisitos, el jefe de analistas debe reaccionar y delimitar los tiempos para modificaciones. Si esto se hace de esta manera, la inserción de nuevas necesidades afectará seriamente al resto de requisitos obtenidos, los mismos que pueden contener inconsistencias y ambigüedades.
32. Problemas introducidos por la etnografía: Los factores sociales y culturales generan prácticas propias entre las personas. Las diferencias sociales, las costumbres, las tradiciones, los idiomas, incluido las economías pueden afectar los sentimientos de los analistas haciendo que estos no logren conceptualizar los requisitos de manera adecuada por lo que es probable una alta introducción de errores en ellos.

1.4 Requerimientos vs requisitos

Según la RAE [3], “requerimiento” significa “Acción y efecto de requerir”, “requerir” significa “necesitar”, y “requisito”, significa “Circunstancia o condición necesaria para algo”; es decir que “requerimiento” es sinónimo de “necesidad” que implica “carencia o falta de algo”. Entonces se deduce que “requerimiento” es la “acción de requerir algo” y “requisito” es el “algo”.

El siguiente ejemplo intenta aclarar el tema: En una organización el gerente general siente la necesidad de contar con información al momento para tomar decisiones rápidas; para lograr ello tiene que someter su necesidad a un conjunto de procesos o normas propias de la organización que permita contar con la información al instante respetando criterios de

estructura. Esta es la definición de “*requerimiento*”.

El *requisito* comienza por entender o comprender de qué forma necesita la presentación de la información solicitada. El gerente general al solicitar la información presenta una petición verbal o por medio de documento escrito; mientras que el analista, para resolver el pedido, tiene que emplear métodos para lograrlo, como por ejemplo “*cubos de datos si se pudiere, caso contrario implementa algoritmos*”.

La discusión sobre esto data de muchos años atrás. Otros piensan que el problema se encuentra en la traducción del idioma inglés, “*request*” significa “*requerimiento*” y “*requirement*” es “*requisito*”; dos términos diferentes siendo el de mayor empleo “*requisito*”. Es necesario aclarar que la palabra “*requirement*”, además de su aceptación valida como traducción de requisito, en inglés es sinónimo de necesidad, lo que genera confusión en el empleo de los términos.

Asimismo, podemos concluir que “*requerimiento*” se refiere a todas las necesidades y deseos pedidos por el cliente y las personas involucradas en la utilización del software, mientras que “*requisito*” está referido a todas las funcionalidades, características, particularidades y restricciones que debería tener el software. Los siguientes ejemplos permitirán tener una idea clara sobre ello.

Ejemplo 1.

Requerimiento: El cliente desea que le construyan un sistema de inventarios para poder conocer lo que tiene en el almacén.

Requisito: El sistema de inventarios debe contener los siguientes módulos: Gestión de usuarios, gestión de productos, gestión de notificaciones, gestión de reportes, gestión de proveedores, gestión de ventas y gestión de facturación.

En el primer caso el cliente solicita la construcción del producto de software porque piensa que le ayudará a resolver sus problemas de información y de toma de decisiones. No indica cómo hacerlo, construirlo o lo que contiene. En el segundo caso, después del análisis practicado, se indica que el sistema debe contener los módulos descritos previamente. Esto implica que existe una relación con la funcionalidad del sistema.

Ejemplo 2.

Requerimiento: El dueño de un conjunto de locales, que los alquila para la celebración de eventos, necesita mantener información sobre la gestión de estos.

Requisito: El sistema estará compuesto por un conjunto de módulos como: Gestión de usuarios, gestión de reserva, gestión de administradores, gestión de locales, gestión de recursos y gestión de reportes.

Un “*requisito*” se puede convertir en un “*requerimiento*”; todo depende del punto de vista

que asumamos. En este ejemplo 2, un usuario puede indicar que dentro del sistema de administración de locales para eventos se necesita un módulo de gestión de reservas; podemos notar que la gestión de reservas tiene un proceso para lograrlo por lo que se convierte en una necesidad, tal como se muestra en el ejemplo 3.

Ejemplo 3.

Requerimiento: El sistema de administración de locales para eventos debe contener un módulo que permita gestionar las reservas.

Requisito: El módulo de gestión de reservas debe contener el registro de información de las reservas, la visualización de la información de las reservas, la modificación de la información de la reserva, así como su eliminación.

Ejemplo 4.

Requerimiento: El dueño de una agencia de turismo solicita mantener actualizada la información de la agencia.

Requisito: El sistema debe contener los siguientes módulos: Gestión de usuarios, gestión de paquetes turísticos, gestión de horario de personal, gestión de operaciones, gestión de turistas, gestión de personal y gestión de inventarios de transporte.

Ejemplo 5.

Requerimiento: El jefe de recursos humanos solicita contar con información del personal que labora en la agencia de turismo.

Requisito: El módulo debe contener la visualización de las valoraciones, capacidad para imprimir la hoja de ruta, así como el contrato de servicio.

Ejemplo 6.

Requerimiento: El administrador solicita se muestre las valoraciones asignadas.

Requisito: El sistema debe mostrar la valorización del personal, la impresión de la hoja de ruta y el contrato de servicio.

Ejemplo 7.

Requerimiento: El jefe del área pide que la hoja de ruta sea impresa.

Requisito: El sistema genera una hoja de ruta por cada grupo de turistas. El formulario se muestra de la siguiente manera:

```
<htable style = "width:100 %>
<tr>
```

```
<th> Código del paquete </th>
<th> Datos del conductor </th>
<th> Datos del vehículo </th>
<th> Datos del Guía </th>
</tr>
</table>
```

Los datos se extraen de la base de datos con el comando:

```
SELECT Codigo_Paquete, Datos_Conductor, Datos_Vehiculo, Datos_Guia FROM Hoja_Ruta.
```

1.5 Clasificación de los requisitos

La literatura muestra clasificaciones donde mezclan los tipos de “*requerimientos*” y los tipos de “*requisitos*”. Wiegers [14] propone una clasificación general definiendo esta de la manera siguiente:

- Requerimientos de negocio: Son aquellos que sustentan la realización del proyecto en función de los beneficios que se esperan alcanzar.
- Requerimientos de usuario: Describen lo que los usuarios pueden hacer con el producto. Esto se integra en función de objetivos y tareas.
- Requerimientos funcionales: Describen lo que el desarrollador debe construir y que se encuentra detallado en la especificación de requisitos de software.
- Requerimientos del sistema: También lo define como “*requerimientos del producto*”. Describen los requisitos de alto nivel observando el sistema como un todo.
- Reglas de negocio: Toma en cuenta las políticas de la organización, directivas del gobierno, regulaciones de la industria y algoritmos computacionales. Estos elementos delimitan el dominio del sistema.
- Atributos de calidad: Define los atributos de calidad que serán tomados en cuenta para la confección del software.
- Interfaces externas: Define la forma como se lleva a cabo la conectividad con fuentes externas al producto de software a construir.
- Restricciones: Las restricciones de diseño e implementación son impuestas al desarrollador por razones de conveniencia de la organización.

La figura 1.1 muestra, según [14], los tipos de requisitos explicados anteriormente, encontrándose asociados a artefactos específicos. Por otro lado, en una edición más reciente, [15] clasifica a los requisitos como “*requisitos de desarrollo*” y “*requisitos de gestión*” mientras que Klaus Pohl [11] clasifica a los requisitos en: requisitos funcionales, requisitos de calidad y restricciones. Asimismo, en Swebok v3.0 [12] y Sommerville [9] solo clasifican a los requisitos

en funcionales y no funcionales. Una clasificación mucho más extensa la proporciona Young [13]; clasifica los requisitos en:

- Requisitos de negocios: Son los requisitos principales que sirven de base para el desarrollo del sistema.

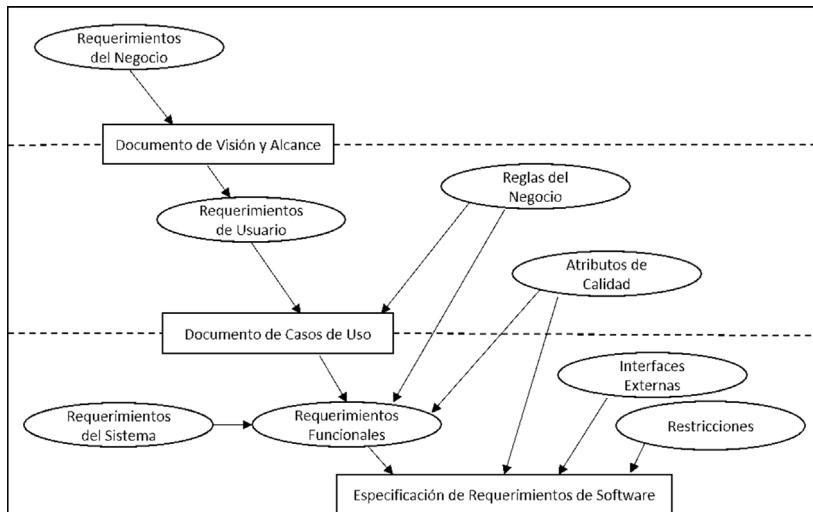


Figura 1.1: Conexión entre los tipos de requisitos según [14]

- Requisitos iniciales: Son aquellos primeros requisitos proporcionados por el cliente; muy genéricos como para formar parte de la funcionalidad.
- Requisitos reales: Requisitos que reflejan las verdaderas necesidades del cliente y que sirven de base para la implementación del sistema.
- Requisitos de usuario: Requisitos proporcionados por los usuarios finales del sistema o por el conjunto de ellos.
- Requisitos de alto nivel: Requisitos que capturan la visión del cliente permitiendo definir el ámbito del sistema.
- Reglas de negocio: Reglas que proporcionan la información base para los requisitos funcionales; es decir las políticas, condiciones y restricciones.
- Requisitos funcionales: Aquellos requisitos que explicitan la funcionalidad total del sistema a implementar.
- Requisitos no funcionales: Aquellos requisitos que no participan en la funcionalidad del producto pero que sus decisiones si lo hacen. Se encuentran representados en términos de calidad por lo que están asociados a los requisitos funcionales.
- Requisitos derivados: Requisitos que se logran producto del refinamiento sucesivo de los requisitos previamente definidos.
- Requisitos de diseño: Requisitos orientados a los artefactos de diseño que guardan

relación con el catálogo de requisitos.

- Restricciones de diseño: Políticas que proporcionan las restricciones necesarias porque complementan la funcionalidad del sistema.
- Requisitos de actuación: Requisitos que guardan dependencia con otros requisitos y que permiten tener visibilidad en lo que respecta a la trazabilidad.
- Requisitos de interfaces: Requisitos orientados a las interfaces gráficas de usuario y a la operación del sistema por parte de los usuarios.
- Requisitos de verificación: Son aquellos requisitos reales que cumplen o satisfacen los criterios de diseño.
- Requisitos de validación: Son todos aquellos requisitos, funcionales o no funcionales que fueron implementados en el sistema.
- Requisitos de calificación: Aquellos requisitos validados o verificados en la aplicación conjuntamente a los elementos de diseño.
- Requisitos de calidad: Son aquellos atributos de calidad que influyen en la determinación de los requisitos y la funcionalidad del sistema. Se encuentran representados en los requisitos no funcionales.
- Requisitos desconocidos: Representan a aquellos requisitos que inicialmente son desconocidos y que a medida que evoluciona la construcción del producto se tornan reales.
- Requisitos del producto: Requisitos que son la respuesta de la ejecución del sistema. Se menciona que representan a la funcionalidad del sistema.
- Requisitos de procesos: Requisitos que son la respuesta a la ejecución de los procesos de desarrollo de software.
- Requisitos de apoyo logístico: Requisitos deducidos del uso de herramientas, capacitaciones, procedimientos e instalaciones.
- Requisitos medioambientales: Requisitos que resultan del entorno físico y de las condiciones sociales y culturales y que influyen como esfuerzo en la construcción del producto.
- Requisitos del sistema: Requisitos asociados con los diferentes niveles del sistema permitiendo implementar las capas necesarias para la construcción del producto.
- Requisitos de los subsistemas: Requisitos relacionados con los subsistemas y su integración total en la funcionalidad general del sistema.
- Requisitos orientados a los componentes: Requisitos generados por la utilización de librerías y componentes o por la reutilización de estos.

No todo lo visto en las clasificaciones se muestran en la realidad, así que tenemos que aprender a ser observadores para poder entender y trabajar sobre una clasificación específica. En otras oportunidades nacen requisitos que, aparentemente, no se encuentran dentro de los marcos de trabajo; para ello debemos tener la visión que permita aglutinar los mismos en

estructuras adecuadas.

En la experiencia de los autores, la clasificación adecuada y que se encuentra en casi todas las aplicaciones desarrolladas o por lo menos en la gran mayoría de ellos se muestran a continuación: la figura 1.2 muestra la clasificación para los requisitos funcionales y la figura 1.3 para los requisitos no funcionales.

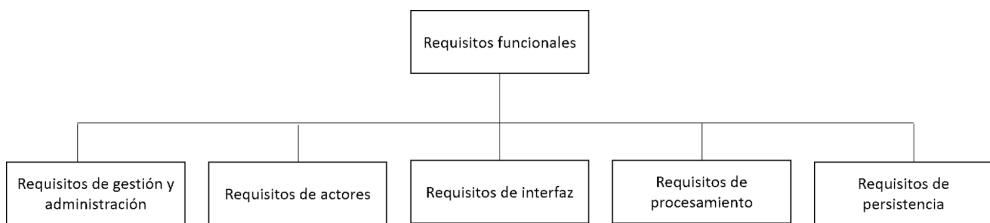


Figura 1.2: Clasificación para requisitos funcionales

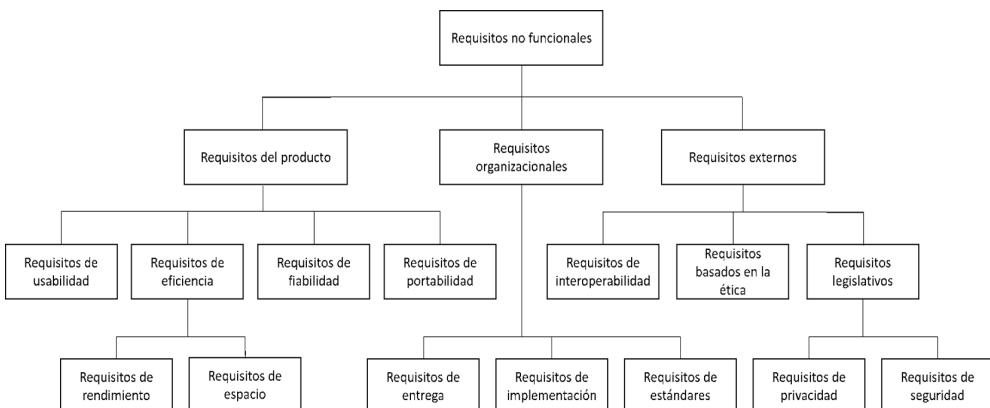


Figura 1.3: Clasificación para requisitos no funcionales

1.6 El problema de la inconsistencia y ambigüedad

De acuerdo con la RAE [3], inconsistencia significa “*falta de consistencia*”, y consistencia significa “*Trabajón, coherencia entre las partículas de una masa o los elementos de un conjunto*”. Esto implica que la inconsistencia en los requisitos es la “*falta de coherencia entre los requisitos que conforman el catálogo*”; es decir, requisitos que se contradigan entre sí dando como resultado la eliminación o complementación de uno de ellos. Por ejemplo:

Ejemplo 1.

Requisito 1. El usuario ingresará a la sección clientes.

Requisito 2. El usuario presionará el botón para ingresar a la sección clientes.

En este caso existe una inconsistencia ya que en el requisito 1 no se sabe cómo ingresar a la sección clientes y en el requisito 2 menciona que se debe presionar un botón para hacerlo, esto no es explícito al contexto. Aquí se debe tomar la decisión de reestructurar el requisito 1 o requisito 2. Como ejemplo reestructuramos el requisito 1 complementando su contenido con elementos del requisito 2.

Requisito 1. El usuario ingresará a la sección clientes por medio de la presión del botón derecho mostrado en el ícono ICO-0014 y que será presentado en el interfaz INT-0006.

Ejemplo 2.

Requisito 1. El sistema presenta el interfaz INT-0018 con los campos 2 y 3 desactivados.

Requisito 2. El usuario puede modificar los datos de los campos mostrados en el interfaz INT-0018.

La inconsistencia se encuentra presente en este par de requisitos. Para resolver este problema, se decide hacerlo en el requisito 2 por lo que la redacción de este requisito queda de la manera siguiente:

Requisito 2. El usuario puede modificar los datos de los campos en el interfaz INT-0018 a excepción de los campos 2 y 3.

Por otro lado, la RAE [3] define a una ambigüedad como “*calidad de ambiguo*” y ambiguo como “*que puede entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión*”. Por consiguiente, ambos términos no son sinónimos ya que la inconsistencia implica una falta de coherencia y la ambigüedad es la múltiple interpretación de un hecho. Ambos adjetivos producen confusión. A continuación, se presentan ejemplos de ambigüedades.

Ejemplo 1.

Requisito ambiguo. El usuario ingresará a la sección clientes.

Para este requisito nos hacemos la siguiente interrogante: ¿Cómo se ingresará a la sección clientes?; la respuesta puede ser por teclado o con un clic en el botón derecho del ratón o con un toque en el interfaz que se muestra en la pantalla o con un toque en el interfaz por medio de un lapicero digital. Se debe recordar que las implementaciones de estos medios son diferenciadas ya que los sistemas operativos reaccionan de manera diferente. La redacción correcta se muestra a continuación:

Requisito no ambiguo. El usuario ingresará a la sección clientes haciendo clic con el botón derecho del ratón en el interfaz INT-0028.

Ejemplo 2.

Requisito ambiguo. El sistema realizará la búsqueda de los datos del cliente.

Este requisito presenta dos claras ambigüedades. La primera responde a la pregunta: ¿Dónde llevará a cabo la búsqueda de los datos del cliente?, y la segunda: ¿Cómo llevará a cabo dicha búsqueda? La primera interrogante puede tener respuestas como: en una estructura de datos o en una base de datos almacenada en algún computador, mientras que para la segunda interrogante puede ser: Por DNI o por apellido paterno o por RUC o por apellido materno o por celular entre otros. Su redacción correcta se muestra a continuación.

Requisito no ambiguo. El sistema realizará la búsqueda de los datos del cliente en la base de datos denominada BD_DATOS por medio del Registro Único del Contribuyente (RUC) que es la llave primaria.

Es importante la solución de las inconsistencias, así como de las ambigüedades ya que su introducción y no corrección pueden acarrear serios problemas a la hora de la codificación del producto de software; los errores detectados en la codificación del producto suelen generar un considerable incremento en los costos de estos. Estos factores son considerados como detractores de la calidad.

1.7 Gestión del cambio en los requisitos

En el momento de la elaboración de los productos de software, la primera fase se dedica a la elaboración del catálogo de requisitos. Se utilizan técnicas para la obtención pensando que la información entregada es la definitiva y que el catálogo de requisitos, una vez concluida la fase de obtención de información, se encuentra lista para su uso en la fase siguiente.

Pocos son los jefes de proyecto que tienen cuidado cuando son solicitados los cambios en los requisitos. Otros soslayan esta actividad pensando que los cambios pueden afectar solo a un requisito o que las modificaciones resultan ser menores. No imaginan que los efectos colaterales pueden llevarse a cabo en cascada en un conjunto de requisitos y que pueden afectar incluso a la fase de codificación.

Los cambios que pueden ser realizados en los requisitos ya sea por una solicitud, por una falta en la apreciación del requisito, por el olvido en los detalles de la información o por descuido de los integrantes del proyecto; pueden causar daños severos en los costos del proyecto o en el tiempo de entrega del producto final. Estos suelen tener efectos en un conjunto de requisitos de allí la importancia de la trazabilidad.

Existen un conjunto de factores que inciden en los cambios de los requisitos y que pueden generar impactos negativos. Es necesario un estricto control de ellos y así poder cuantificar

el efecto hacia otros requisitos. Mitigarlos es tarea ardua que consume recursos que pueden desbalancear el aspecto económico del proyecto, pero deben ser resueltos para sustentar los criterios de calidad asumidos. Se presentan algunos de ellos:

1. **Cambios tecnológicos:** Los cambios tecnológicos se presentan todos los días por lo que las necesidades de los clientes pueden ser tratados de diferentes formas con tecnologías diferentes.
2. **Cambio en las estrategias del negocio:** Es común comenzar la aplicación con unas estrategias y los clientes se encuentran trabajando en otras; estos desbalances influyen en la construcción de los aplicativos.
3. **Cambios en las prioridades del negocio:** Como la toma de información para la formulación de requisitos se hace en una fecha determinada es probable que se encuentren elucubrando otras prioridades que no son transmitidas al equipo de desarrollo.
4. **Modificaciones de las leyes:** Un caso típico son los cambios de valores en los impuestos los mismos que son previstos por organizaciones externas a la empresa que desarrolla la aplicación.
5. **Por un mal enfoque en los requisitos no funcionales:** Es bastante común que los analistas no tomen en cuenta inicialmente estos requisitos ya que se dedican íntegramente al desarrollo del aplicativo. Un ejemplo común es desarrollar un aplicativo standalone cuando el cliente lo necesita vía web.
6. **Por una mala apreciación de los requisitos funcionales:** La inexperiencia de los analistas hace que no entiendan las necesidades reales del cliente. Esto conlleva a un desperdicio de recursos.
7. **Por un mal enfoque del analista:** Sigue lo mismo con los analistas; si no tienen la experiencia del caso entonces no podrán tener un enfoque correcto del modelo del negocio a automatizar.
8. **Por una mala planificación y toma de información en las entrevistas:** Las entrevistas deben ser planificadas de manera que se oriente a objetivos específicos. Si las entrevistas se llevan a cabo de manera informal entonces no se sabrá qué preguntar en su oportunidad.
9. **Cambios en la visión del sistema que ha sido planificado:** Normalmente el cliente traslada sus necesidades a los desarrolladores los mismos que automatizan sus procesos; pero los analistas jamás preguntan sobre sus visiones de futuro con respecto al modelo del negocio.
10. **Cambio en la percepción de los usuarios:** Los usuarios son los que tienen la experiencia producto de su trabajo en el día a día y por lo tanto no precisan lo que realmente desean. Esto afecta a la construcción de la aplicación.
11. **Cambios producidos por el ambiente del negocio:** Los modelos de negocios son dinámicos y este dinamismo se convierte en un factor que en muchas oportunidades se transforma en elemento negativo para la construcción del correspondiente aplicativo.
12. **Cambios del mercado donde se desenvuelve el negocio:** El comportamiento del mercado

también produce efectos negativos en la construcción de la aplicación ya que los analistas tienen una percepción del modelo en un tiempo dado.

Los cambios deben controlarse y documentarse, se convive con el cambio. Por lo tanto, es esencial planificar posibles cambios a los requisitos cuando el sistema sea desarrollado y utilizado. Luego, la gestión de los requisitos de software, en su evolución, es un proceso externo que ocurre a lo largo del ciclo de vida de la construcción del software y del proyecto, este debe ser controlado en su integridad.

La gestión de requisitos es el conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y sus cambios. Es decir, consiste en gestionar los cambios a los requisitos acordados, las relaciones entre ellos, las dependencias entre la educación, ilación y especificación de requisitos juntamente con otros documentos producidos por el proceso de desarrollo de software.

Esta actividad asegura la consistencia entre los requisitos y los criterios de calidad asumidos dando como resultado el consumo de grandes cantidades de tiempo y esfuerzo. La gestión de cambios involucra actividades como: establecer políticas, guardar históricos de cada requisito (control del historial), identificar dependencias entre ellos y mantener un control de sus versiones.

Es vital planificar los cambios de acuerdo con las prioridades que los clientes determinen durante la identificación. Se deben aprobar los mecanismos para la configuración del cambio, así como las políticas de trazabilidad. Las actividades inherentes a la gestión del cambio deben ser planificadas de tal forma que tengan como fin el logro de los objetivos y propósitos; esto se logra gracias a una eficiente comunicación.

Según [16][17] mencionan que la gestión del cambio en los proyectos es una actividad que debe ser planificada con la finalidad de alcanzar el objetivo propuesto. Además, indican que *"El proceso de control de cambios tiene una secuencia de pasos que son los ideales a seguir para poder definir cuáles serán los cambios que aplicarán en el desarrollo del sistema, los pasos por los que se transita serán el análisis de la solicitud, valoración del cambio, análisis de modificaciones, y la documentación del cambio una vez que este ha sido aceptado y desarrollado en el proyecto."*. La figura 1.4 muestra el proceso de cambio.

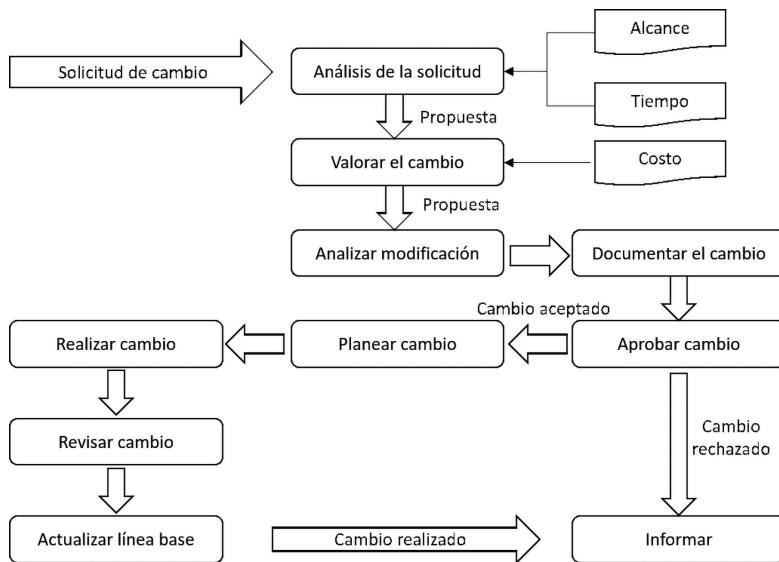


Figura 1.4: Proceso de cambio. Fuente [16]

1.8 Herramientas de soporte

Las herramientas de soporte constituyen un importante activo para la elaboración del catálogo de requisitos. Ayudan a simplificar tareas con el objetivo de alcanzar información consistente; pero presentan la desventaja de que se encuentran orientadas a sólo una parte del proceso. Estas son algunas de las herramientas.

1. REM. Herramienta experimental que sólo ejecuta la fase de Ingeniería de Requisitos de un proyecto de software. Este es un producto gratuito basado en la metodología "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información" de la tesis doctoral del Dr. Amador Durán Toro [18].

Presenta cuatro artefactos: Documento de Requisitos del Sistema, Documento de Análisis del Sistema, Registro de Conflictos y Defectos y Registro de Peticiones de Cambio. La trazabilidad la lleva a cabo mediante una matriz donde la variable principal son los objetivos. La figura 1.5 muestra su interfaz.

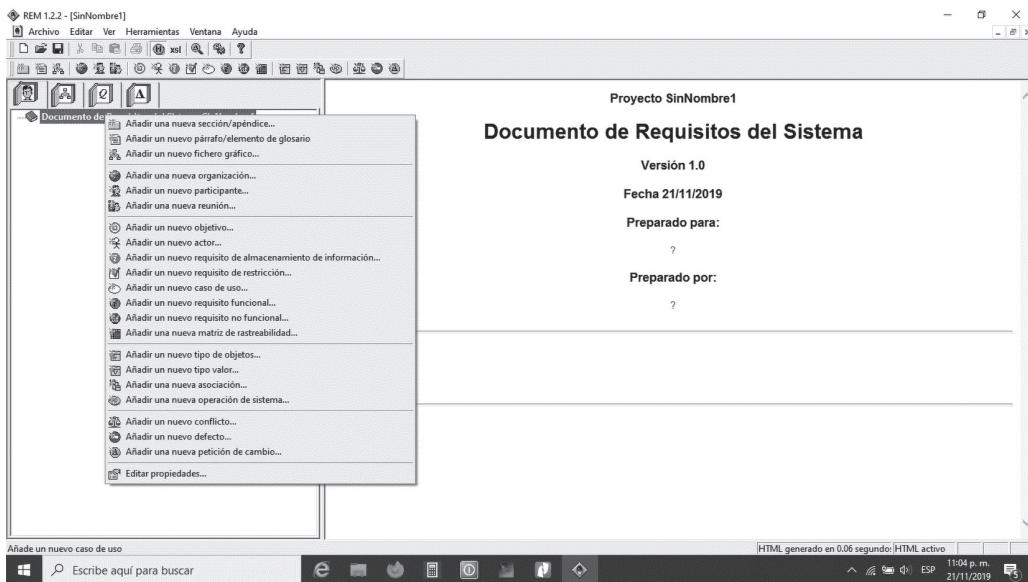


Figura 1.5: Interfaz de REM. Fuente [18]

2. Heler. Herramienta libre para la especificación de requisitos que ofrece soporte a las actividades de Ingeniería de Requisitos contempladas en la fase de entendimiento del problema, enmarcada dentro del Proceso Unificado. Contiene módulos de proyecto, stakeholder, actores y casos de uso así como el de requisitos [19]. La figura 1.6 muestra la interfaz.



Figura 1.6: Interfaz de Heler. Fuente [19]

3. Rational Requisite Pro. Facilita la revisión de requisitos en un entorno de equipo del proyecto. Se puede organizar y acceder a toda la documentación del proyecto desde una única ubicación y, a continuación, los miembros del equipo pueden compartir comentarios sobre requisitos específicos o aspectos más amplios del proyecto a través de discusiones en línea.

Todos los elementos de discusión se almacenan en la base de datos de proyecto para su posterior revisión. Para revisar documentos de requisitos, un autor puede proteger, temporalmente, un documento durante la revisión. Más tarde, las revisiones se fusionan en el proyecto y se ponen a disposición de todos los miembros del equipo [20]. Su interfaz se presenta en la figura 1.7.

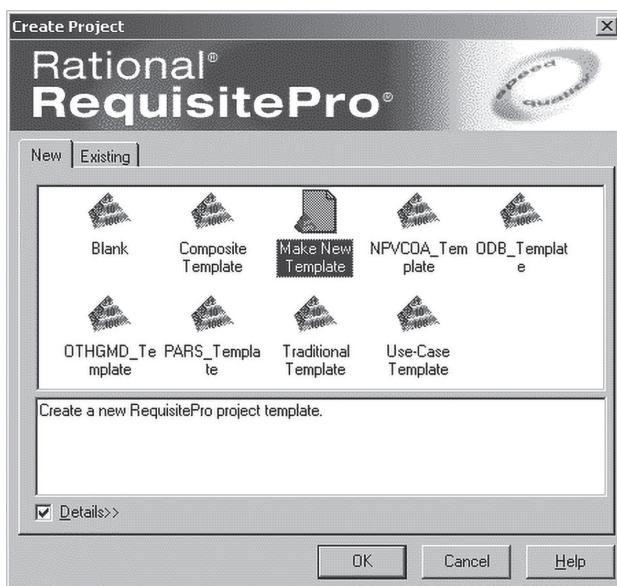


Figura 1.7: Interfaz de IBM Rational RequisitePro. Fuente [20]

4. OpenOME. Es una herramienta de análisis y modelado general orientada a objetivos y/o agentes. Proporciona a los usuarios una interfaz gráfica para desarrollar modelos y admite el acceso a una poderosa base de conocimientos que permite un sofisticado análisis asistido por computadora. Esta herramienta está destinada a proporcionar a los desarrolladores de software un vínculo claro entre los requisitos, las especificaciones y las fases de diseño arquitectónico del desarrollo.

También se está prestando atención en el uso de esta herramienta para la reingeniería de procesos de negocios. OpenOME integra una versión mejorada de OME con otras herramientas para admitir la ingeniería de requisitos orientada a objetivos, agentes y aspectos en el desarrollo de software, el modelado conceptual y otros entornos de edición de gráficos [21]. La figura 1.8 presenta el interfaz.

Este kit de herramientas puede ser empleada con facilidad si el usuario tiene la experiencia del caso. Por ejemplos, el marco de los requisitos no funcionales deben estar expresados en expresiones regulares, y estos son poco común para ser empleados por usuarios externos.

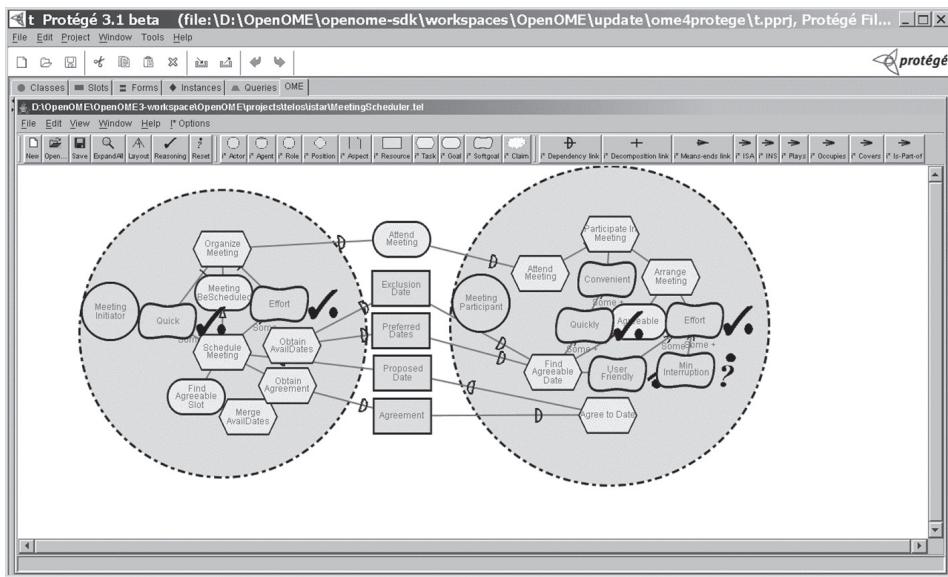


Figura 1.8: Interfaz de OpenOME. Fuente [21]

5. OSRMT. Es una herramienta de Software Libre pensada para asistir en todo el Ciclo de Vida del Desarrollo del Software. Permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba).

La herramienta integra módulos de Administración y Configuración, Gestión de Documentos de la Ingeniería de Requisitos, Trazabilidad entre documentos de trabajo e Informes y estadísticas. También es posible personalizar los atributos de las funcionalidades, requisitos, casos de prueba, pudiendo configurar valores por defecto para los atributos, y personalizar las vistas [22]. Su interfaz es mostrada en la figura 1.9.

El módulo de administración es independiente al de configuración ya que en el primero solo se dedica a llevar los requisitos y realizar cambios necesarios mientras que en el segundo configura los mismos para establecer patrones de trabajo. La trazabilidad que lleva a cabo es entre la documentación más no entre artefactos de construcción del producto de software

Visure Requirements es una herramienta completa de ingeniería de requisitos, flexible, probada, capaz de agilizar el proceso de requisitos de software como parte del proceso de definición mecánica y de hardware. Visure Requirements ayuda a la colaboración efectiva en proyectos y aumenta la calidad del software a través de la captura, análisis, especificación, validación y verificación, gestión y reutilización de requisitos. Etapas que son consumidas complementadas con responsabilidad y mesura.

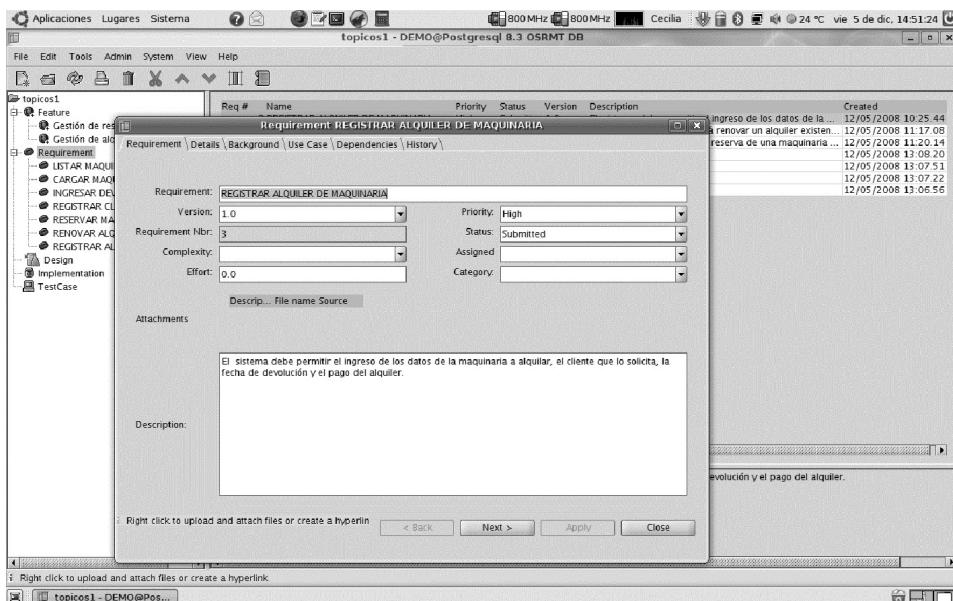


Figura 1.9: Interfaz de OSRMT. Fuente [22]

6. RETools. La mayoría de las herramientas de modelado admiten solo una notación (o como máximo algunas notaciones), lo que impide que los analistas utilicen las notaciones más apropiadas para la tarea de modelado particular. RE-Tools es un kit de herramientas de código abierto implementado utilizando un perfil UML para StarUML, también una herramienta de modelado UML de código abierto.

El kit de herramientas admite muchas notaciones de modelado de requisitos principales, incluido el Marco NFR, el Marco i*, KAOS, Marcos de problemas y UML. Cada una de estas notaciones puede usarse para modelar diagramas correspondientes independientes o junto con requisitos no funcionales (NFR). El kit de herramientas también admite el razonamiento cualitativo original del Marco NFR y aumentado con uno cuantitativo [23]. Su interfaz se muestra en la figura 1.10

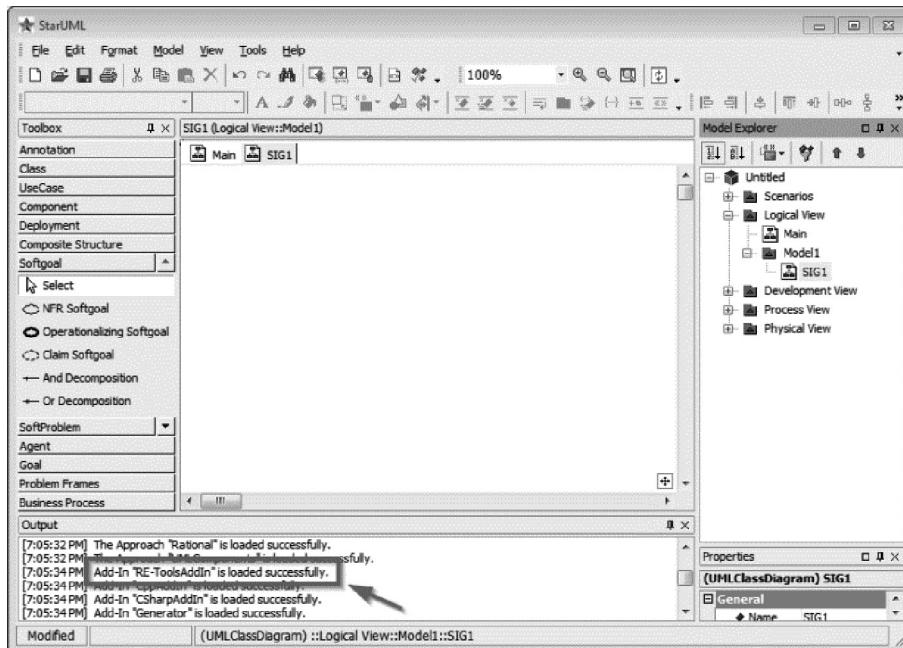


Figura 1.10: Interfaz de RETools. Fuente [23]

7. Reto UPV. Creada en el año 2004 en la Universidad Politécnica de Valencia (España). Versión académica de la herramienta RETO-UPV versión 2.0 (Requirements Engineering Tool). Esta permite la definición de la misión del sistema, la construcción del árbol de refinamiento de funciones y el desarrollo del modelo de casos de uso [19].
8. Visure Requirements. Visure Requirements es una solución completa y flexible de Ingeniería de Requisitos capaz de gestionar eficientemente sus procesos de requisitos permitiendo la colaboración de forma más eficaz, favoreciendo la calidad de sus productos; y dando soporte completo a la captura, análisis, especificación, validación y verificación, gestión y reutilización de los requisitos [24]. La figura 1.11 muestra una interfaz.

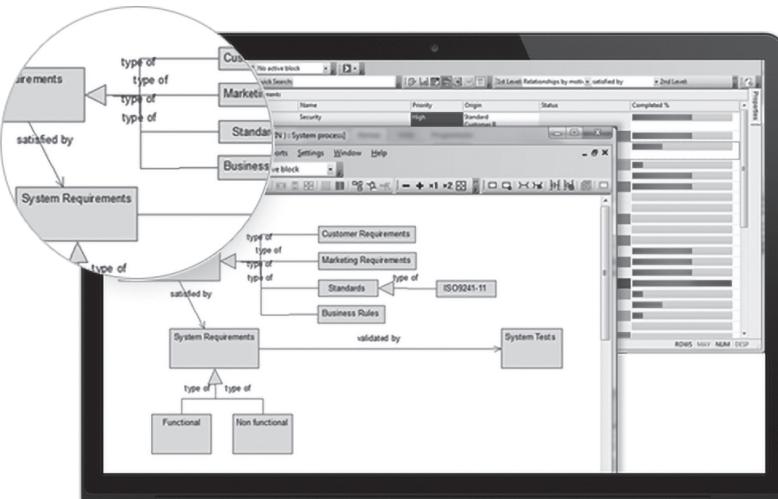


Figura 1.11: Interfaz de Visure Requirements. Fuente [24]

- IBM Rational DOORs. DOORs(Dynamic Object Oriented Requirements System) es un software para la gestión de requisitos creado por la empresa sueca Telelogic, y adquirida por IBM. Es una base de datos que permite el almacenamiento estructurado y la administración de requisitos, que cuentan con atributos. Para permitir el seguimiento de los requisitos durante el tránsito del proyecto se pueden enlazar los objetos entre ellos. Cuenta con una serie de herramientas para el análisis y permite automatizar diferentes pasos en el uso de la herramienta con la ayuda de un lenguaje de script llamado DXL (DOORS eXtended Language)[25]. La figura 1.12 muestra una interfaz del producto.

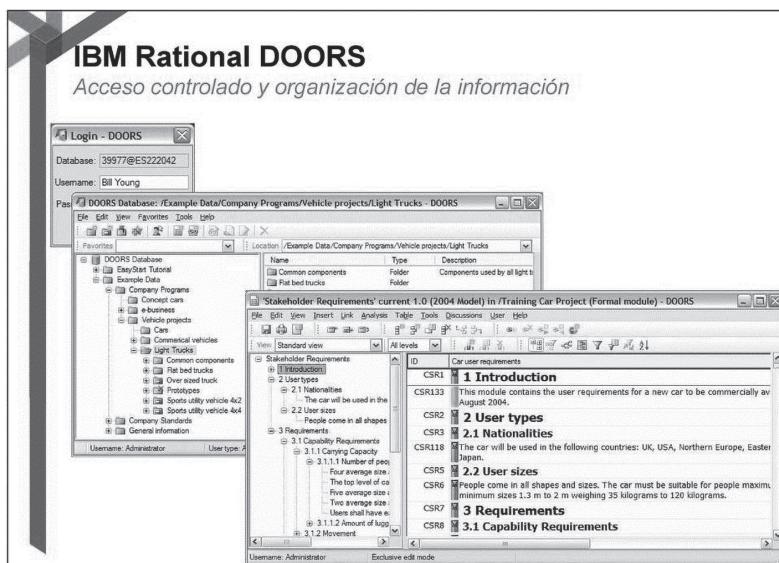


Figura 1.12: Interfaz de IBM Rational DOORs. Fuente [25]

10. Reqify. Es una aplicación interactiva de fácil uso para la gestión de requisitos, trazabilidad y análisis de impacto entre diferentes sistemas, programas y niveles de proyectos en todo el ciclo de vida de desarrollo del software y el hardware. Vincula los procesos de verificación y desarrollo a los requisitos. Ayuda a mantener los equipos del proyecto centrados en labores de implementación y verificación, para obtener la máxima eficiencia en el desarrollo de sistemas integrados complejos [26].

Ofrece una completa lista de interfaces con varias herramientas de ingeniería de sistemas. Reqify puede capturar datos desde cualquier fuente de cualquier proveedor en una amplia variedad de formatos de datos y archivos. Es una plataforma abierta y ampliable que cuenta con interfaces de más de 60 herramientas habituales de ingeniería de sistemas. También permite a las organizaciones gestionar eficazmente sus procesos de ingeniería de requisitos y garantizar el cumplimiento de normas como ISO61508, ISO26262, Spice, D0178C, D0254, FDA, GAMP, CMMI, entre otros [26]. La figura 1.13 muestra una de sus interfaces.

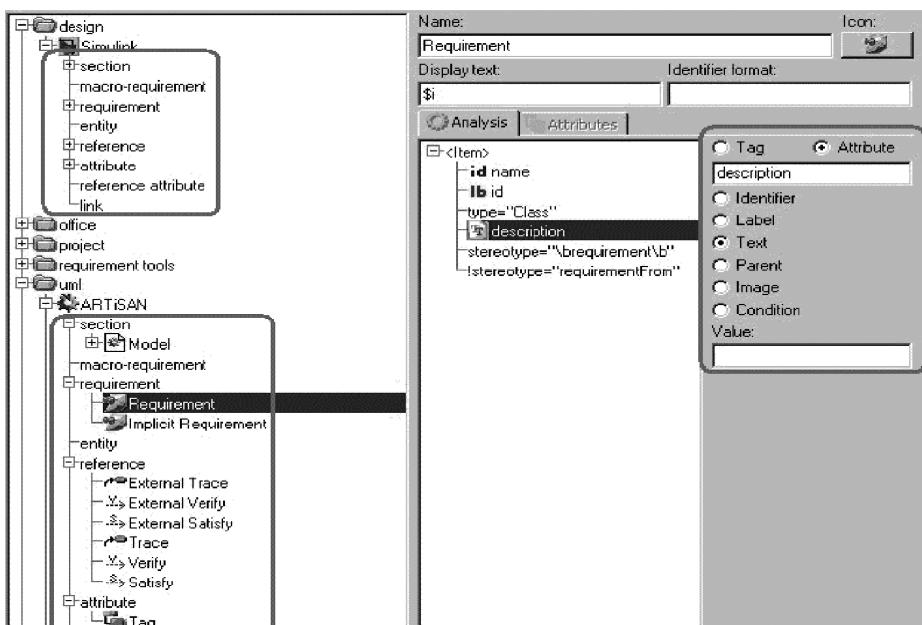


Figura 1.13: Interfaz de Reqify. Fuente [26]

11. Jama. Software orientado a la definición, gestión, verificación y validación de requisitos. Puede obtener buenos requisitos de la manera más rápida, desde el inicio hasta la producción. Los equipos de ingeniería de sistemas que desarrollan productos críticos para la vida y la economía utilizan este producto con la finalidad de innovar a partir de las limitaciones de entornos altamente complejos [27]. La figura 1.14 muestra un interfaz.

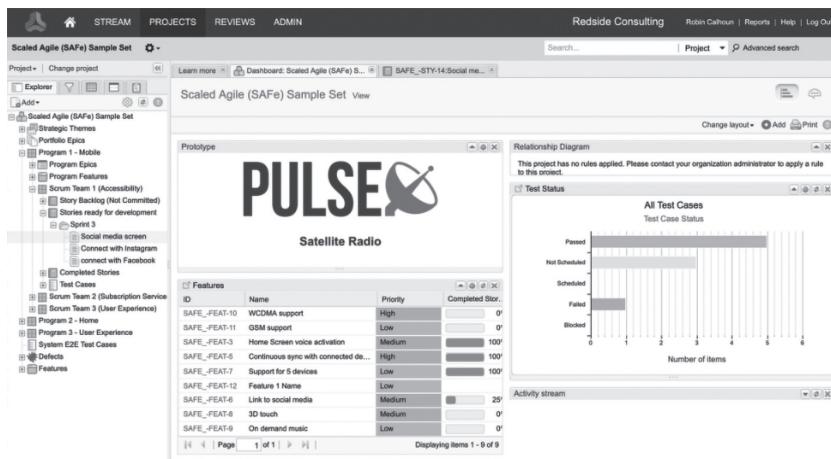


Figura 1.14: Interfaz de Jama. Fuente [27]

12. Accept 360. Herramienta donde la estrategia de aceptación y la gestión del portafolio de productos simplifican lo que se necesita para desarrollar e implementar estrategias efectivas que maximicen las oportunidades de mercado. También ayuda a las organizaciones a garantizar que los objetivos comerciales se conserven a medida que los productos pasan de los conceptos iniciales a la ejecución final. Presenta una estrecha relación con la gestión de ideas y la gestión de requisitos priorizando la comunicación de información con la planificación de productos.
13. Gatherspace. Software creado por la empresa Gatherspace.com para la gestión de requisitos. La gestión de requisitos los hace en línea empleando una herramienta orientada a los casos de uso para modelar los requisitos. Asimismo, ofrece un conjunto de artefactos que proveen una adecuada ayuda en línea, así como permitir explicitar las necesidades del usuario [28]. La figura 1.15 muestra su interfaz principal.

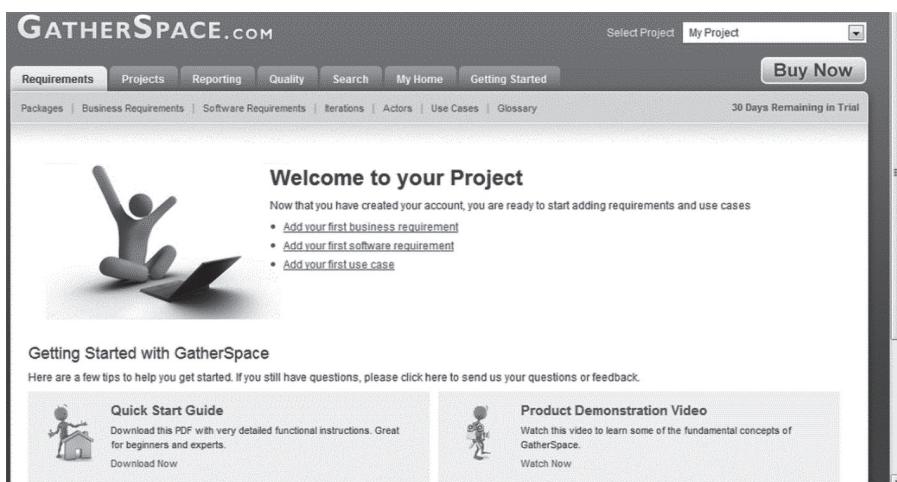


Figura 1.15: Interfaz de Gatherspace. Fuente [28]

14. MagicDraw. Herramienta CASE desarrollada por No Magic. Compatible con el estándar UML 2.3, desarrolla código para diversos lenguajes de programación, así como para modelar datos. Cuenta con capacidad para trabajar en equipo y es compatible con varios entornos de desarrollo. Permite el desarrollo colaborativo directamente con la herramienta a través del Team Work Server (Software que permite trabajar a más de un desarrollador sobre el mismo proyecto en un mismo instante, el modelo está almacenado en un equipo servidor y los desarrolladores pueden consultar y actualizar la información) [29]. La figura 1.16 muestra su interfaz principal.

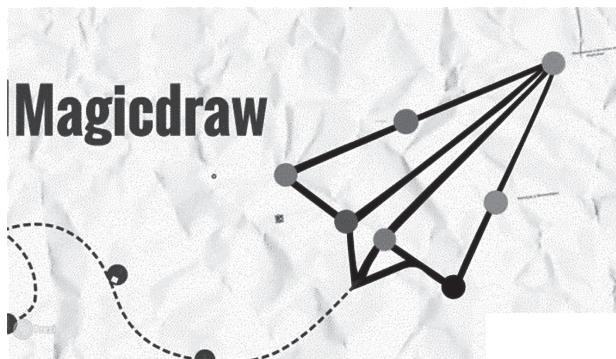


Figura 1.16: Interfaz de MagicDraw. Fuente [29]

15. CaliberRM. Herramienta que es útil para la construcción de sistemas grandes y complejos y proporciona una base de datos de requisitos con trazabilidad. La organización ve a los requisitos como parte del proceso de gestión de la calidad del software, al igual que, las pruebas (testing) y el trazado de defectos (defect tracking). Caliber está basado en internet y maneja referencia de documentos, responsabilidad de usuario, trazabilidad, prioridad y estado entre otras características [30]. La figura 1.17 muestra un interfaz de esta herramienta.

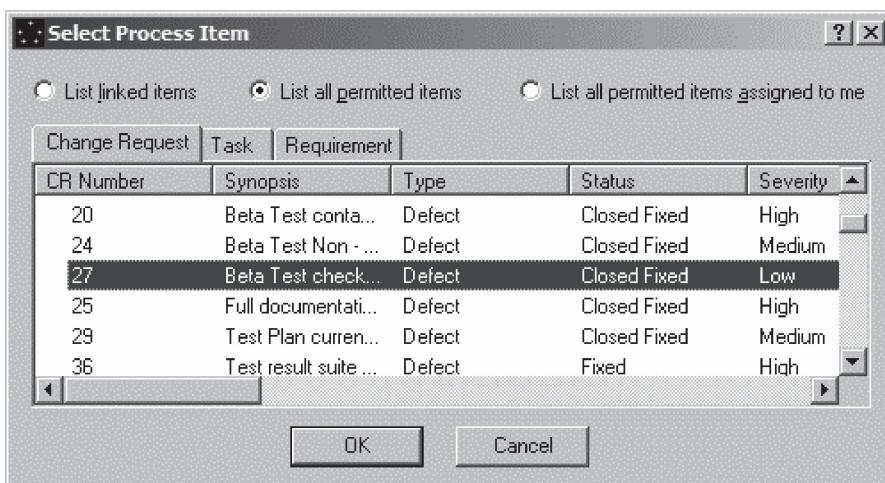


Figura 1.17: Interfaz de CaliberRM. Fuente [30]

16. Integral Requisite Analyzer. Herramienta desarrollada que soporta los procesos de Ingeniería de Requisitos de una metodología propietaria definida por dicha empresa para la Especificación de Requisitos. Consiste en un entorno gráfico y textual que permite generar una documentación sencilla que los clientes pueden revisar en las primeras fases de construcción de una solución. IRqA además, posee módulos para la definición y gestión de pruebas de aceptación, estimación de costos basados en casos de uso, gestión de versiones y navegación por los dominios del negocio. El inconveniente con esta herramienta es su escasa flexibilidad para adaptarse a metodologías operativamente diferentes para la cual fue desarrollada [31].
17. Reto. Herramienta que propone un modelo de requisitos para capturar los aspectos funcionales del sistema; básicamente, mediante tres técnicas complementarias entre sí: la definición de la Misión del Sistema, la construcción del Árbol de Refinamiento de Funciones y el desarrollo del Modelo de Casos de Uso. Además, se introduce un Proceso de Análisis que permite traducir el Modelo de Requisitos en el Modelo Conceptual, manteniendo la trazabilidad entre ambos y propiciando una representación de la información en el segundo prototipo. No hace mención sobre el tratamiento de los requisitos no funcionales [32]. La figura 1.18 muestra un ejemplo de su interfaz.

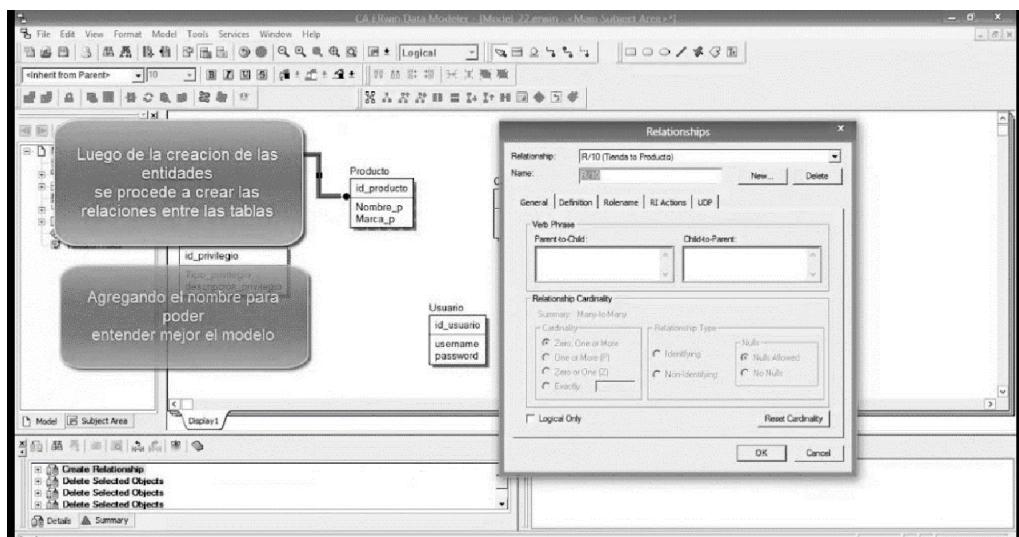


Figura 1.18: Interfaz de RETO

18. Controla. Herramienta de apoyo al proceso de desarrollo de software orientado a las pequeñas organizaciones, con enfoque en la administración de requisitos. Permite la identificación de los requisitos cercano al stakeholder, sus detalles y la administración de los cambios a través de la rastreabilidad y el control de versiones. El control lo lleva a cabo por medio de matrices de trazabilidad que permiten introducir la planificación de las pruebas de software siempre que se tome en cuenta el riesgo correspondiente [33].

19. Jeremía. Se trata exclusivamente de una aplicación cliente, la cual no permite la posibilidad de trabajar en equipo. Ayuda durante el desarrollo del sistema, especialmente en el seguimiento de cambios de los requisitos a lo largo del ciclo de vida. Es posible captar las necesidades, analizarlas y clasificarlas. Implementa un módulo orientado a la generación de la documentación posible de exportar en formato DocBook XML, la cual, junto con los requisitos, se almacena en una base de datos en MySQL [32]

CAPÍTULO 2

Elicitación de Requisitos

2.1 Introducción

El término “*elicitación*” no se encuentra definido en uno de los diccionarios de habla hispana más importantes como lo es la Real Academia de la Lengua, pero aparece en el Diccionario Panhispánico de Dudas 2005. Este término es una adaptación del verbo inglés: “*to elicit*”, y se exhibe en los textos de psicología. En este contexto *elicitar* significa “*provocar, suscitar u obtener*”.

En el ámbito de la Ingeniería de Requerimientos, el término *elicitación* cobra relevancia ya que se trata de recolectar y organizar información referente al sistema y que conlleva al entendimiento de las necesidades del cliente en función del modelo de negocio a automatizar. La *elicitación* implica un modelo de la organización de los requerimientos para lograr construir un producto de software con criterios de calidad.

2.2 Definición

La *elicitación* de requerimientos es el proceso de revelar los requerimientos para la construcción de un sistema por medio de una efectiva comunicación con los clientes, usuarios del sistema y stakeholders. Es de suponer que los actores guardan una relación estrecha con el producto a construir, es decir, que practican la gestión del conocimiento relacionadas con tareas, actividades y procesos del modelo de negocio.

R. Muelas [34] hace referencia a “...*puesta en marcha de técnicas que sirven para recopilar conocimiento o información sobre personas. Por tanto, la elicitation no es más que una técnica usada para conseguir información de forma discreta, sin que la persona se entere. Esto hace que sea necesario conseguir la información directamente de la persona que la posee*

D. Cohn [35] manifiesta que "*La elicitation de requerimientos es una de las principales tareas que debe llevarse a cabo para la correcta implementación de un desarrollo de software. Su incorrecta especificación genera costos innecesarios a lo largo del proyecto e inclusive, su completo fracaso*". Esto implica el especial cuidado que se debe tener con la información adquirida para construir un producto de software.

Finalmente, podemos decir que también la elicitation (que proviene del latín "elicitus", que significa "inducido" y "elicere" que significa "atrapar") se refiere a la transferencia de información de manera fluida entre seres humanos teniendo como medio al lenguaje. La interrelación puede ser variada ya que la información puede ser trasladada de un sistema a otro, de un computador a una persona o de persona a persona.

2.3 Objetivo de la elicitation

Elicitar un requerimiento implica que se debe analizar, inspeccionar y concebir una postura que debe ser resuelta, así como una necesidad que se espera sea cubierta además de lograr una adecuada funcionalidad. Toda la información tiene que ser entendida, traducida y documentada para que cuando se haga el mantenimiento respectivo este debe ser entendido. Es decir, que los integrantes del proyecto de software; analistas, diseñadores, programadores, arquitectos de software, evaluadores de software, entre otros, al leer los requerimientos puedan proyectar su solución.

Así pues, la elicitation de requerimientos es un proceso que tiene como objetivo principal el de entender las necesidades del cliente y transformarlas en esquemas de diseño. Esta interacción se basa en la comunicación efectiva con los actores del sistema. Esta tarea es compleja ya que se debe de emplear un conjunto de técnicas y ciclos de evaluación, incluidas técnicas etnográficas, para construir productos de calidad.

2.4 Problemas de la elicitation

Los problemas que se suscitan durante la elicitation de requerimientos se muestran a continuación:

1. Problemas suscitados por el lenguaje

A menudo ocurren hechos que no se deben dejar de lado ya que los requerimientos son obtenidos de manera verbal o escrita. Cuando la elicitation se lleva a cabo de manera verbal es común que los analistas no entiendan la intensidad de transmisión de la información por lo que a menudo tergiversan el contenido. La mala comprensión o el mal entendimiento de los requerimientos conducen a una pésima conceptualización de los mismos.

Cuando los requerimientos son escritos entonces puede que el texto oculte errores sintácticos y semánticos siendo más graves los segundos. Estos generan un mal entendimiento o incomprendición del requerimiento y como consecuencia de ello se introduzcan errores en las fases subsiguientes de la construcción del producto de software.

2. Problemas suscitados por la etnografía

El aspecto y nivel cultural de las personas producto de pertenecer a una etnografía también introducen errores en los requerimientos. Si el cliente solicita la construcción de un producto de software y este pertenece a una zona quechua puede suceder que el analista no domine este idioma y por lo tanto no pueda eliciar correctamente los requerimientos.

Las expresiones producto de las manifestaciones culturales como lo son costumbres, mitos entre otros pueden confundir a los analistas ya que estos, al no pertenecer al contexto, pueden concebir una interpretación errónea de los requerimientos. Los localismos o regionalismos son ejemplos claros de estos malos entendidos.

3. Problemas suscitados por la documentación sustentatoria

En vista que los documentos son artefactos que dependen de la redacción de las personas que elaboran estos, también se pueden detectar errores léxicos, sintácticos y semánticos. El mayor problema está en artefactos cuyos diseños se encuentran en contradicción con los procesos de la organización o la existencia de resoluciones que dejan sin efecto parte de ellos o con cambios sustanciales que ameritan un vistazo a los procesos que los soportan. De ser así pueden causar efectos colaterales a otros requerimientos.

4. Problemas suscitados por la reutilización de requisitos

En innumerables oportunidades estamos tentados en usar lo que ya habíamos resuelto con algún otro producto de software, cuando se trata de componentes es fácil de resolver, pero cuando se trata de requerimientos se debe tener cuidado en que lo que se extrae resuelva exactamente el proceso de la organización en donde se construye el producto de software.

5. Problemas suscitados por la reutilización de artefactos de software

En otras oportunidades se tiende a reusar diagramas de casos de uso, secuencia, objetos, clases, procesos, colaboración entre otros; y cambiar solo aquellos que se piensa resuelve el problema. Cuando esto se lleva a cabo es probable que la modificación leve en la reutilización de un artefacto pueda que no conduzca a la actualización del requerimiento y si este depende o influye sobre otros el error puede fluir sobre un conjunto de requerimientos.

6. Problemas suscitados por la utilización de herramientas de software

Cuando encontramos herramientas de software que apoyan a la gestión o administración de requerimientos, siempre estas tienden a resolver la forma de pensar del autor o autores. Es probable que una herramienta defina directamente lalicitación sobre la base de requerimientos funcionales o no funcionales sin considerar las complejidades del sistema a construir. Es necesario que los analistas adecuen las herramientas a sus necesidades, pero si alguno de ellos conoce las complejidades resulta siendo favorable construir sus propias herramientas.

7. Problemas suscitados por la mala estructuración del catálogo de requisitos

Es común hallar que el catálogo de requisitos se encuentra conformado por una lista enumerada de requisitos funcionales y no funcionales y sin llevar el control de versiones o la historia de cada uno de ellos. Cuando se produce un cambio, es necesario que se reescriba en el mismo catálogo y en esencia porque se pierden los historiales de cambio. Para ello es indispensable que el analista estructure el catálogo de requisitos, pero pensando en un artefacto que permita resolver el problema.

8. Problemas suscitados por la falta de experiencia en lalicitación de requerimientos

Normalmente cuando se trata de tomar u obtener información sobre requerimientos, los jefes de proyectos envían a los analistas junior; al no contar con la experiencia del caso no toman en cuenta técnicas, métodos o metodologías que les pueden servir de ayuda. Respuesta a ello tienden a demorar en la comprensión de la lógica del negocio y a redactar requerimientos inexactos introduciendo inconsistencias o ambigüedades en ellos.

9. Problemas suscitados por el cambio de personal del proyecto de desarrollo de software

Otro de los problemas es el cambio de personal después de haber conseguido desarrollar un conjunto de requerimientos. Estos cambios producen que el personal reemplazante pierda el tiempo en entender el modelo del negocio para luego enfrentarse en la determinación de los requerimientos sin conocer cómo es su efecto colateral, es decir la influencia sobre otros requerimientos.

10. Problemas suscitados por el mal modelamiento del sistema

Cuando no se logra entender de manera adecuada el modelo del negocio se tiende a construir el catálogo de requisitos bajo una falsa percepción. El modelamiento inadecuado infiere una mala comprensión de lo que se desea construir y puede introducir inconsistencias y ambigüedades en los requisitos colaterales.

11. Problemas suscitados por la falta de retroalimentación en los requerimientos

En muchas oportunidades los artefactos son diseñados y versionados sin retroalimentar esas diferencias en los requerimientos base. Este efecto puede producir una percepción equivocada cuando por ejemplo se lleve la codificación del producto a construir y sin corregir a otros artefactos que se encuentran asociados al artefacto construido.

2.5 Funciones de la elicitation

La ingeniería de software proporciona las herramientas necesarias para construir software bajo el punto de vista de calidad. Una de estas herramientas es la Ingeniería de Requerimientos que tiene relación con actividades que ayudan a comprender las necesidades de los clientes para luego llevarlas a una estructura adecuada con la finalidad de lograr una codificación estable.

Para lograr ello se hace indispensable emplear técnicas o metodologías que permitan lograr una adecuada elicitation de requisitos que permitan plasmar la construcción de estos productos de software; construcción que tiene como primera fuente los requerimientos en su forma de entender las necesidades del cliente y lograr una codificación adecuada del mismo bajo los lineamientos de algún lenguaje de programación. Las funciones de la elicitation se muestran a continuación:

1. Extraer información correspondiente al modelo del negocio y los sistemas involucrados

Esta es una de las primeras tareas que se debe llevar a cabo. Conocer adecuadamente el modelo del negocio implica conocer los procesos que soportan la organización vistos desde el ángulo organizacional y operacional. Asimismo, se trata de entender como los sistemas actuales resuelven la problemática de la organización en función de sus objetivos y cuál es el flujo de información con la que trabajan. Esta tarea es importante porque la comprensión de esta situación implica saber lo que se desea hacer.

2. Planificar las reuniones de negociación

Las reuniones de negociación son entrevistas entre el analista y el cliente y tienen por finalidad eliciar la información respecto al sistema a construir. Cuando se trata de un problema de construcción de software se debe poner un alto interés en la planificación de las entrevistas con el cliente. Los autores hacen hincapié en la forma como se debe de llevar a cabo estas entrevistas las mismas que deben ser precisas y con periodos de tiempo bastante cortos.

3. Reconocer las finalidades del sistema

Los analistas deben entender por qué se desea construir un producto de software y los motivos deben guardar relación con lo que desea la organización. Los motivos deben ser

explícitos para que el producto de software a construir resuelva los problemas reales de la organización obedeciendo y respetando la esencia de los procesos y el flujo de información que administra la organización.

4. Reconocer los requerimientos de información

Entendido el modelo de negocio de la organización se procede a obtener y reconocer todos los requerimientos de información necesarios para resolver el problema de automatización. Los requisitos deben estar ordenados y estructurados de tal manera que no contengan inconsistencias y ambigüedades. Su ordenamiento implica un control de la trazabilidad de los mismos permitiendo conocer que las necesidades del cliente sean representadas en las interfaces gráficas de usuario.

5. Reconocer los requisitos funcionales

Entre el conjunto de requisitos identificados producto de la información obtenida, se deben de identificar los requisitos funcionales que son aquellos que presentan o guardan una relación directa con las funcionalidades del sistema. Los requisitos funcionales deben reflejar el comportamiento o las particularidades funcionales del sistema a automatizar cuando se respetan ciertas condiciones.

6. Reconocer los requisitos no funcionales

No solamente los requisitos funcionales cobran importancia en el afán de construir un sistema. Son considerados importantes los requisitos no funcionales ya que estos pueden ocasionar serios defectos colaterales en la construcción del producto de software. Atributos como la eficiencia, seguridad, dependencia y usabilidad del sistema deben ser considerados en la construcción del mismo.

2.6 Técnicas de la elicitation

A continuación, se presenta un resumen de las técnicas de elicitation propuestas por autores en diferentes trabajos de investigación.

1. Cuestionarios

Los cuestionarios son documentos donde se plasman un conjunto de preguntas asociadas con la resolución de un problema en particular. La elicitation se apoya en este artefacto para obtener importante información pormenorizada; una excesiva cantidad de preguntas puede provocar fallas en las respuestas mientras que una pequeña cantidad de ellas puede terminar ocultando información que no podrá ser tomada como requerimiento.

2. Encuestas

Las encuestas son estudios detallados que consisten en la recopilación de información para después analizarla. A partir de este análisis es que se pueden determinar los requisitos para la automatización de un sistema además de las condicionantes para que los mismos expresen las reales necesidades del cliente. Este análisis permite reconocer las inconsistencias o ambigüedades de la información.

3. Entrevistas

Las entrevistas son mucho más flexibles para obtener información, pero presentan el inconveniente del lenguaje de comunicación. Estas diferencias permiten que no se comprenda lo que se desea haciendo factible la introducción, con suma facilidad, de inconsistencias y ambigüedades. La expresividad es otro factor que introduce elementos no deseables en la comprensión de la información.

4. Análisis de documentos

Los requisitos pueden ser obtenidos del conjunto de documentos que mantiene la organización como componente principal del quehacer diario. Por ejemplo, los procesos se encuentran en el artefacto denominado: Flujo de Procesos; también el flujo de información puede encontrarse representada en los documentos presentados por usuarios finales desde mesa de partes.

5. Lluvia de ideas

Esta técnica implica reuniones con los especialistas en requisitos para poder elaborar estrategias que permitan obtener información que conforman el requisito. La lluvia de ideas permite definir criterios para obtener información o para saber si la información existente mantiene las características deseadas para elaborar un buen catálogo de requisitos permitiendo su clasificación de manera adecuada.

6. Grupo de discusión (Focus group)

Técnica que permite recopilar información sobre aspectos cualitativos asociados a los requisitos funcionales y no funcionales; generalmente se reúne un grupo pequeño de integrantes que conforman el proyecto con el objetivo de socializar opiniones referentes a las preferencias del cliente y así poder estructurar el sistema a construir bajo el punto de vista de la descomposición.

7. Desarrollo rápido de aplicaciones (RAD) / Diseño de aplicación conjunta (JAD)

RAD (Rapid Application Development) es un proceso de desarrollo de software que se encuentra orientado a la interacción con el usuario final, de manera imprescindible, vía la construcción de prototipos. JAD (Joint Application Design) es un proceso de diseño de

software que tiene como principal prioridad reunir requerimientos para automatizar los sistemas de información.

8. Prototipado

Esta técnica comienza por el modelado rápido de un producto de software entregando una solución que tiene fortalezas en determinadas características. Su exposición permite aprender, probar y validar para luego agregar los requisitos funcionales como capas sólidas en la construcción del producto final. De esta manera se genera una satisfacción del cliente producto de la visualización de resultados confiables.

9. Métodos basados en objetivos

Técnica que proporciona un modelo claro del tipo de información que se desea recolectar y el mismo es empleado para guiar toda la actividad de elicitation de requerimientos incluido los requisitos no funcionales. El modelo contempla la estructuración de la información y la forma como serán obtenidos los requerimientos funcionales, es decir que provee las técnicas adecuadas para eliciar los mismos.

10. Métodos basados en escenarios

Esta técnica consiste en que previamente se debe de tener conocimiento sobre el sistema a automatizar y su relación con los objetivos a alcanzar. La técnica permite la implementación de la modularidad ya que admite, en función de las necesidades del cliente, limitar el dominio del problema. Para lograr eficiencia en la aplicación de la técnica se debe confeccionar las descripciones de una situación a resolver y un camino de hechos que logren el dinamismo de la información.

11. Escalonamiento

Técnica que permite la conectividad entre los analistas y la información de las áreas a automatizar relacionados con los productos a obtener. Se puede emplear otras técnicas para extraer información, la misma que es consultada con los usuarios finales logrando que la distribución de la misma quede a lo largo del escalonamiento permitiendo conjeturas sobre los requisitos.

12. Clasificación de tarjetas

Técnica que en primer lugar permite observar el comportamiento de los usuarios del sistema a construir con respecto a sus puntos de vistas del uso de la información para luego discernir entre los puntos de vista de los usuarios y lograr recepcionar los mejores requerimientos permitiendo, de esta manera, lograr una concepción adecuada de las necesidades del cliente y de la organización.

13. Repertorio de rejillas

Técnica que identifica la forma como un usuario interpreta la forma de administrar la información incluida su estructura organizativa. Su flexibilidad permite emplear enfoques cualitativos y cuantitativos. Para aplicar esta técnica es importante que el analista tenga claro el modelo del negocio para poder reconocer los flujos de información y su tratamiento futuro dentro del contexto de los requisitos.

14. Métodos etnográficos

Técnica que permite el estudio social de los integrantes de las áreas en estudio y la forma como estos entregan los requisitos para la construcción de un producto de software. El comportamiento social, incluida su interrelación, debe ser tratado con cuidado para poder eliciar de manera correcta las necesidades del cliente asociadas con los objetivos de la organización. Es común observar lo que ocurre para luego solicitar explicaciones e interpretaciones sobre la información.

15. Etnometodología

Técnica que se basa en el supuesto de que todos los seres humanos tienen un sentido práctico con el cual adecuan las normas de acuerdo con una racionalidad práctica que utilizan en la vida cotidiana. Es decir que trata sobre el estudio de la forma como el usuario observa los requisitos entregando opiniones y reglas de juego personales que les permite resolver el problema del consumo de información.

16. Análisis de conversación

Técnica que implica un severo análisis de las conversaciones sostenidas con el cliente, actores o stakeholders con la finalidad de obtener patrones para luego introducirlas en cualquiera de las etapas de lalicitación de requerimientos; tiene como finalidad el ahorro de tiempo en la confección del catálogo de requisitos debido a la exposición de un conjunto de premisas.

2.7 Proceso de la elicitation

El proceso de elicitation atraviesa tres fases; la fase de educación de requisitos que genera el artefacto de educación y desde el cual se puede obtener el modelo conceptual; la fase de ilación de requisitos que genera el artefacto de ilación pudiendo obtener el modelo del diseño del sistema y la fase de especificación que genera el artefacto de especificación de requisitos y desde donde se genera el modelo de codificación del sistema.

Estas tres fases componen los requisitos funcionales y que juntamente con los requisitos no funcionales, ingresan al proceso de validación retroalimentando el control de versiones

de las fases producto y la corrección de inconsistencias o ambigüedades. Luego de este procedimiento, el último control de versiones de los artefactos conforma el catálogo de requisitos. La figura 2.1 muestra el proceso.

El proceso de elicitation implica la toma y procesamiento de información para convertirlos en requisitos. Para ello existen un conjunto de técnicas que permiten mantener un rastro importante en la información entregada por el cliente. Por otro lado, es común que esta información sea tomada de manera informal ya que la documentación de los mismos no es una actividad que las empresas desarrolladoras la usen.

Es bueno entender, que la fase de educación se encuentra orientada a tomar la información del cliente; la fase de ilación implica el entendimiento y transformación de la información por parte del analista; y la fase de especificación el traslado de la información en niveles primarios de la codificación.

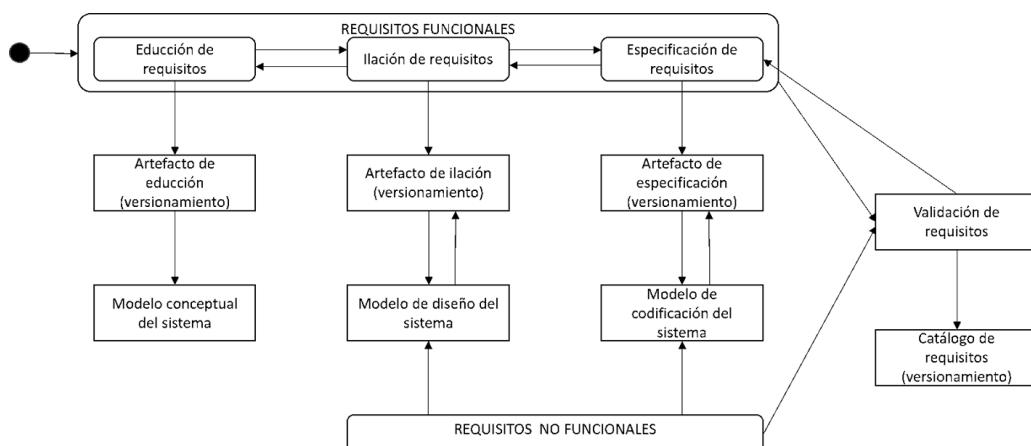


Figura 2.1: Proceso de elicitation de requisitos

De la figura 2.1 se desprende que el proceso de elicitation contempla tres fases que permite la construcción del catálogo de requisitos: Educación, ilación y especificación. La fase de educación se encuentra relacionada con el cliente o stakeholders, la fase de ilación con la conceptualización del modelo del negocio por parte del ingeniero de software [36]; y la fase de especificación con la primera etapa de la codificación del producto final.

2.8 Metodologías para solicitar requisitos

A pesar de la existencia de varias metodologías para solicitar requisitos, una completa exposición se muestra en M. Torrente & E. Piriutella [37]. A continuación, se muestra un resumen de las metodologías investigadas.

1. Casos de uso perdidos (Misuses cases)

Metodología que también es conocida como “casos de abuso” y que permite identificar requisitos de seguridad; otras metodologías se centran en la obtención de requisitos funcionales.

2. Sistemas blandos (Soft Systems)

Metodología que resuelve el problema de la identificación de requisitos en función de factores sociales y políticos. Ejemplos de este tipo de sistemas son: como solucionar la falta de insumos en un sistema de stocks o cuando existen problemas sociales que afectan la funcionalidad del aplicativo.

3. Despliegue de la función de calidad (Quality Function Deployment)

Metodología que es empleada como medio para interpretar las necesidades del cliente en requisitos orientados a resolver problemas durante el ciclo de vida de construcción de productos de software. También se encuentra orientado a la priorización de requisitos eliminando inconsistencias de los mismos.

4. Expresión controlada de requisitos (Controlled Requirements Expression)

Metodología que permite analizar y especificar requisitos desde el punto de vista del usuario final y cuyos resultados son expresados como flujos de datos estructurados. Definido como un método de madurez permitiendo flexibilidad para eliciar los requisitos y para implementar una revisión incremental de flujos de información y de actividades de procesamiento.

5. Sistema de información basado en problemas (Issue Based Information System)

Metodología que se basa en el intercambio de información de las partes interesadas. Formalmente se intercambian experiencias y perspectivas personales con la finalidad de resolver los problemas de diseño centrándose en la articulación de elementos claves del mismo, permitiendo desmenuzar las posiciones personales con relación a la solución del problema.

6. Desarrollo de aplicaciones conjuntas (Joint Application Development)

Metodología orientada al desarrollo de sistemas grandes y complejos, cuyo objetivo primordial es el involucramiento de todas las partes interesadas en la solución del problema por medio de reuniones de trabajo estructuradas. En la etapa de Ingeniería de Requerimientos, el equipo se encarga de recopilar hechos e información orientados, específicamente, a la seguridad.

7. Análisis de dominio orientado a la funcionalidad (Feature Oriented Domain Analysis)

Metodología centrada en el análisis del dominio proporcionando una descripción genérica de los requisitos para esa clase de sistemas. La forma que adopta es la del modelo de dominio obteniendo un conjunto de criterios para su implementación. Es común emplear FODA para obtener las abstracciones y refinamientos correspondientes.

8. Análisis crítico del discurso (Critical Discourse Analysis)

Metodología que emplea métodos sociolingüísticos para analizar la información obtenida en forma verbal o por escrito. En particular, se puede emplear para analizar las entrevistas delicitación de requerimientos y para comprender las narrativas e “historias” que surgen durante ellas.

9. Método de requisitos acelerados (Accelerated Requirements Method)

Metodología que facilita la descripción y elicitación de requisitos, para lograr ello, el Ingeniero de Software debe llevar a cabo una preparación adecuada de las sesiones de trabajo para que posteriormente un facilitador conduzca a los participantes en la educación de requisitos; finalmente se difunden los resultados como un grupo de requisitos.

2.9 Relación con los estándares

Existen relaciones entre estándares y modelos para llevar a cabo la elicitación de requisitos. El estándar ISO/IEC 26702 es una guía para la aplicación y gestión de los procesos de la ingeniería de sistemas, y el estándar IEEE Std. 1233 que es otra guía para el desarrollo de especificaciones de requisitos de sistemas.

La ISO/IEC 26702. (2007), y la IEEE Std. 1220-2005 contemplan aspectos que tratan exclusivamente con la ingeniería de requisitos. El estándar IEEE Std. 1233(1998), es la guía para el desarrollo de especificaciones de requerimientos de sistemas de la IEEE, en el que se da la pauta para el desarrollo de un conjunto de requerimientos que satisfacerán una necesidad específica. Al conjunto de requerimientos se le llama Especificación de Requerimientos de Sistema (System Requirements Specification, SyRS).

La IEEE Std. 1233(1998) hace hincapié del trabajo con los clientes; los analistas filtran entradas y extraen el grupo de requerimientos, establecen requerimientos derivados necesarios y crean los mismos. Es un proceso iterativo con la meta de extraer todos los requerimientos del sistema y asegurarse que los requerimientos no se encuentran repetidos y que no existan faltantes.

También se encuentran los modelos UP (Unified Process), RUP (Rational Unified Process), PSP (Personal Software Process) y TSP (Team Software Process) que se utilizan como guía para

ejecutar proyectos enfocados a la ingeniería de productos de software cobrando relevancia la ingeniería de requisitos.

Ninguno de los estándares vistos trata una forma ordenada de conservar los requisitos de software, simplemente hacen mención del trato de los requerimientos y requisitos y la documentación que los hacen sostenibles.

CAPÍTULO 3

El Catálogo de Requisitos

3.1 Introducción

Según la Real Academia Española, “catálogo” significa “*Relación ordenada en la que se incluyen o describen de forma individual libros, documentos, personas, objetos, etc., que están relacionados entre sí*” por lo que el catálogo de requisitos implica una relación ordenada de requisitos, los mismos que guardan una relación con respecto a un objetivo general.

A pesar de esta definición, los desarrolladores de software solo logran la elicitation de requisitos bosquejando un único documento donde plasman los mismos bajo un único criterio, y en donde no se logra detectar la etapa a la que pertenece esta actividad. Por tanto, no existe claridad si los requisitos pueden ser codificables, trazables o educionados; esto hace que en la etapa de trazabilidad no se encuentre la correspondiente asociatividad permitiendo que las pruebas de software puedan llevarse a cabo como una actividad independiente de los requisitos.

El catálogo de requisitos debe ser un documento totalmente ordenado en donde los requisitos guardan una interrelación entre sí para lograr el objetivo trazado. El ordenamiento implica una organización del mismo de forma que la trazabilidad pueda llevarse a cabo en cualquiera de las etapas. Estas etapas se encuentran asociadas a los requerimientos expresados por el cliente, a los entendibles por los analistas o los codificadores del producto.

Por otro lado, la organización de este artefacto permitirá que sean hallados, de manera rápida y sutil, los requisitos relacionados con los problemas inherentes al análisis, diseño y codificación del software con respecto a la arquitectura del software definida. Estos hallazgos permitirán correcciones rápidas en un tiempo reducido.

3.2 Definición

El catálogo de requisitos es un documento, que forma parte de la documentación técnica, donde los requisitos se encuentran ordenados, organizados y relacionados con el objetivo general para que a partir del mismo se obtenga la arquitectura de software, la codificación y el plan de pruebas del producto construido.

La organización de este documento es importante ya que sirve, en el futuro, como material de apoyo a los Ingenieros de Software que llevan a cabo el mantenimiento del producto. La simplicidad de organización del catálogo de requisitos permite comprender las fortalezas y debilidades del producto.

3.3 Estructura del catálogo de requisitos

El catálogo de requisitos consta de ocho secciones las mismas que se describen a continuación:

1. Información de la organización: área donde se almacena información importante de la organización que solicita el producto, así como la información del que construye el producto. Es importante porque permite una fluida comunicación entre clientes, analistas, diseñadores y codificadores del producto.
2. Información de los actores: área donde se almacena la información de las personas que cumplen un rol dentro del uso del producto, así como en la funcionalidad del mismo. Los actores presentan una codificación adecuada que permite mantener la hoja de ruta de los requisitos y de su consumo final.
3. Información de los autores: área donde se especifica información de las personas que desarrollan el producto de software; la finalidad de ello es conocer el rastro de quienes hacen los requerimientos o requisitos.
4. Información de las fuentes: área donde se almacena la información de las fuentes que suministran información para la construcción del software. Su especificación implica documentos de estudio, solicitudes, memorándums, oficios y manuales entre otros.
5. Información de la educación de requisitos: área dedicada a la educación de los requisitos y plasmados en una plantilla definida por el jefe del proyecto. Su especificación se encuentra orientada a la opinión de los clientes tal como piensan que funcionan los procesos en la organización.
6. Información de la ilación de requisitos: área donde se almacena la ilación de requisitos permitiendo el entendimiento del modelo del negocio por parte del Ingeniero de Software. Su traducción debe ser clara y coherente de tal forma que la siguiente sección no sufra las consecuencias.
7. Información de la especificación de requisitos: área donde se almacena la información de los requisitos, pero traducidas en los primeros aspectos formales de algún lenguaje

de programación y en forma genérica permitiendo la facilidad para traducirla en el futuro con la consiguiente ganancia de tiempo.

8. Información de los requisitos no funcionales: área donde se consignan los requisitos no funcionales que se encuentran asociados con el producto afectando a la funcionalidad del mismo. Estos requisitos guardan una estricta correspondencia con el análisis y el diseño del producto de software a construir.
9. Información referida a otros aspectos de la construcción del software como por ejemplo las métricas de software, las pruebas de software y la gestión del riesgo de los requisitos.

Esta estructura puede ser variada de acuerdo a los intereses de la construcción del producto, de la metodología o modelo empleado por parte del jefe de proyecto, pero para fines de orientación, se suele elaborar de manera ordenada. La experiencia del constructor permite adecuar el conjunto de plantillas necesarias donde queda plasmada la correspondiente información. La figura 3.1 muestra un esquema de su organización expresado en término del Lenguaje de Modelado Unificado.

El conjunto de plantillas se encuentran diseñadas bajo el concepto de casos de uso. Para ello se han agregado características que son soportados por los casos de uso pero que al mismo tiempo le dan la versatilidad para poder representar información adecuada de los casos previstos, permitiendo que los constructores de software puedan agregar su propia visión de experiencia.

Dependiendo del tipo de software a desarrollar (tiempo real, distribuido o standalone) puede variar la estructura de las plantillas y de los casos de uso. Obviamente que las metodologías empleadas influyen en la modificación de estas plantillas proporcionando solidez a futuras soluciones de los problemas que intentan resolver.

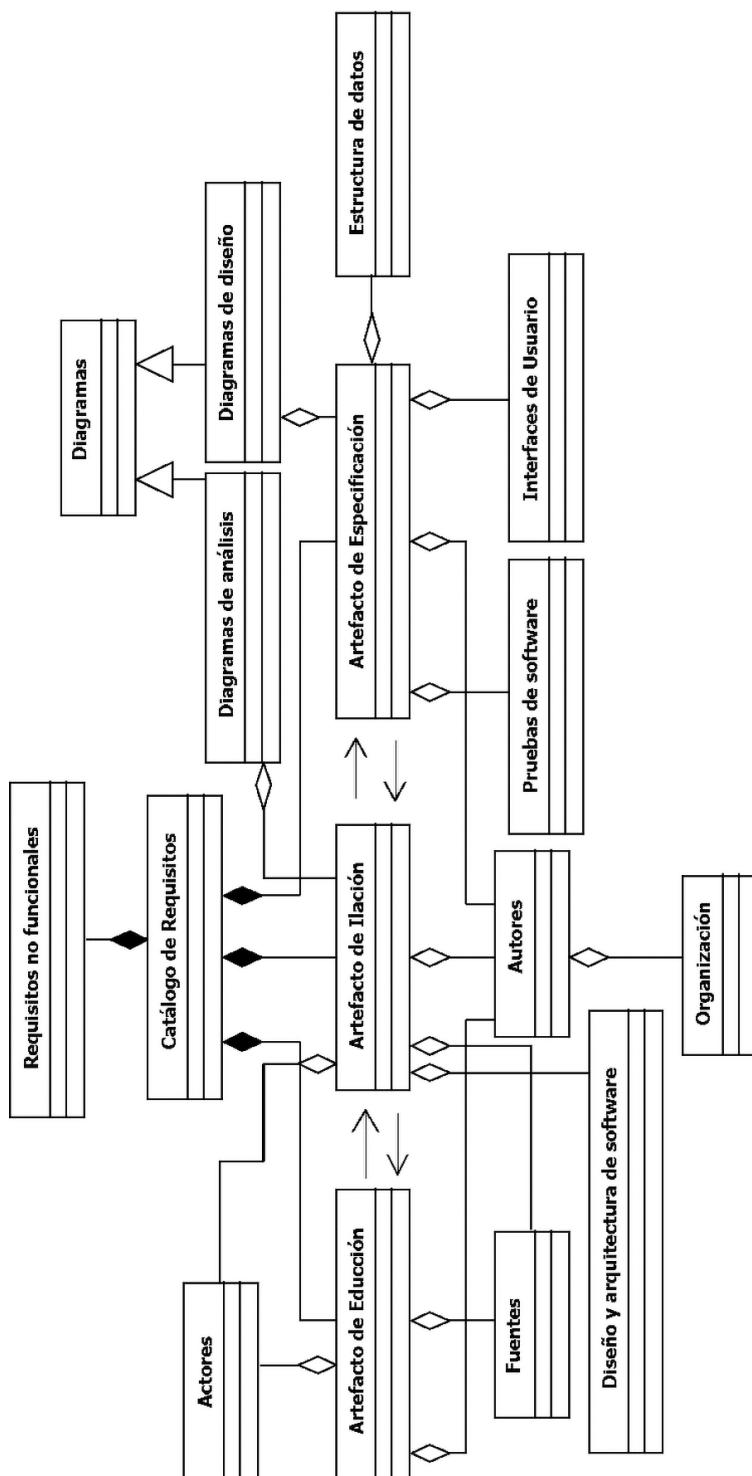


Figura 3.1: Estructura del catálogo de requisitos

3.4 Artefactos de apoyo al catálogo de requisitos

Existen varias herramientas de software que apoyan a la organización de los requisitos producto de lalicitación de los mismos. Estas presentan orientaciones diferentes, aunque tienen varios puntos en común. La tabla 3.1 muestra una relación de estas, así como su principal objetivo.

Herramientas	Descripción
Rational Requisite Pro	Herramienta orientada a los documentos, lleva a cabo el control de cambios de los requisitos incluida su trazabilidad.
CaliberRM	Empleada para la gestión de requisitos de sistemas complejos y de gran tamaño. Analiza los requisitos desde el punto de vista de la calidad.
Integral Requisite Analyzer	Una de las herramientas construidas más completas del mercado, una de sus principales características es la relación de requisitos.
Telelogic Doors	Herramienta multiplataforma que permite la captura, trazabilidad, enlazado, análisis y manejo de cambios en los requisitos.
Requisite Management	Sistema que genera un documento normalizado para el catálogo de requisitos. Lleva a cabo una organización adecuada de los requisitos en donde la trazabilidad es la actividad de mayor importancia.
DRES	Orientado a grupos de trabajo que diseñan sistemas distribuidos permitiendo modificaciones por medio de internet.
OSRMT	Herramienta orientada a documentar el ciclo de vida del desarrollo de software. Define requerimientos derivados.
IBM Rational DOORS	Software que permite la captura, rastreo, análisis y gestión del cambio de los requisitos.
Visure Requirements	Herramienta de última generación que entrega soporte integral al proceso de requisitos. Incluye la captura, análisis, especificación, validación y verificación de requisitos incluida su respectiva trazabilidad.
Reqifly	Software orientado a la administración de la trazabilidad y el impacto del análisis de los requisitos.
Jama	Software que permite definir una buena calidad en los requisitos. Define, administra, verifica y valida requisitos apropiados para la codificación de un producto de software.
Accept 360	Software que contiene como uno de sus módulos la administración de requisitos. La información es administrada para proyectos integrales.
Gatherspace	Software orientado a los negocios dando énfasis a la administración de información para la generación y administración de requisitos. Orientado a métodos ágiles.
MagicDraw	Herramienta CASE orientada al desarrollo de código fuente en diferentes lenguajes de programación. Tiene integrada la herramienta CaliberRM.

Tabla 3.1: Herramientas de software que apoyan a la gestión de requisitos

3.5 Plantillas de trabajo

A continuación, se realiza una descripción de las plantillas de trabajo diseñadas, como base, para agrupar la correspondiente información.

3.0.1 Organizaciones

Esta plantilla almacena información importante con respecto a las organizaciones involucradas en el desarrollo del software, se deben de consignar tantas plantillas como organizaciones intervengan en la solución del problema. Si un módulo es resuelto por otra organización como producto de una tercerización, también debe insertarse la información correspondiente. La tabla 3.2 muestra su diseño, donde:

1. Código: Implica una codificación asignada a la organización: Esta consta de cuatro dígitos anteponiéndole el prefijo ORG-. Su formato depende de los criterios del jefe de proyecto o de los jefes de grupos para la continuidad de todo el proyecto. Ejemplo ORG-0001 u ORG-0006.
2. Versión: Es la versión de la tabla que consta de dos dígitos seguido por un punto y finalizando en dos dígitos. Por ejemplo, V12.23 significa la versión 12.23 y donde los dos primeros dígitos significan cambios drásticos en la información y los subsiguientes, cambios menores. El control de versiones permite realizar un seguimiento histórico de la información proporcionada.
3. Nombre: Es el nombre de la organización. En esta situación se generan dos o más plantillas, una para la organización solicitante y otra para la desarrolladora. Se debe tener en cuenta que se describen las organizaciones solicitantes y las que desarrollan el producto de software. Por ejemplo, SOFTWARE SOLUTIONS S.A.C.
4. Dirección: Es la dirección física donde se encuentran establecidas las organizaciones relacionadas con la construcción del producto de software. También es importante anotar las páginas web correspondientes y las direcciones de las sucursales con que cuentan las organizaciones.
5. Teléfono de la organización: Es el número de teléfono de la organización. Puede ser un número fijo o un número de celular. Es importante contar con números que sean atendidos directamente por los representantes ya que en algún momento se deben tratar decisiones importantes en los requisitos.
6. Fecha: Es la fecha donde comienza la construcción del producto de software o la fecha de llenado de la plantilla correspondiente. Las fechas deben ser correlativas de tal forma que se tiene continuidad en la planificación del proyecto y las actividades relacionadas con la construcción de los módulos. El formato debe ser definido por el jefe de proyecto y ser coherente durante la elaboración de todos los artefactos seleccionados.
7. Representante legal: Se consigna el o los nombres y apellidos de la persona que representa

a la organización y que hace frente a la construcción del producto de software. Puede ser el Gerente General, Gerente de Informática, Gerente de Negocios, Gerente de Personal u otra persona consignada para tal fin.

8. Teléfono del representante legal: Número de teléfono del representante legal.
9. Contacto: Se consigna el o los nombres y apellidos de la persona que funge de contacto en la organización solicitante. Si se trata de una organización que desarrolla el producto se consigna el o los nombres del jefe del Proyecto anotando cualquier observación que se viere por conveniente.
10. Teléfono del contacto: Número de teléfono del contacto.
11. Tipo: Puede ser contratado o contratante. En ambos casos debe ser llenada una plantilla.
12. Autor: Código de la persona que lleva a cabo lalicitación. Esto es: AUT-DDDD.
13. Estado: Permite conocer la situación de la plantilla pudiendo ser de tipo pendiente o concluido.
14. Comentario: Redacción de comentarios si los hubiere.

Atributos	Descripción
Código	ORG-DDDD
Versión	VDD.DD
Nombre	
Dirección	
Teléfono organización	
Fecha	
Representante legal	
Teléfono del representante legal	
Contacto	
Teléfono del contacto	
Tipo	
Autor	
Estado	
Comentario	

Tabla 3.2: Plantilla para la organización

3.0.2 Actores

La plantilla se encuentra diseñada para mantener información de los actores que intervienen en el análisis y diseño del sistema. La tabla 3.3 expone el diseño de la plantilla cuyos campos significan lo siguiente:

1. Código del actor: Es el código asignado al actor; consta de las iniciales ACT seguido de un valor correlativo de cuatro dígitos. Ejemplo: ACT-0023.
2. Versión: Versión de la tabla, consta de dos dígitos seguido por un punto y finalizando en dos dígitos. Por ejemplo, V1.12.
3. Rol: Es el rol que desempeña el actor durante el proceso de utilización del producto de software. Estos pueden ser: analista, desarrollador, evaluador de software, gerente de ventas, gerente de personal. Un actor puede cumplir varios roles para lo cual se deberán generar sus tablas correspondientes.
4. Organización: Código de la organización a la que pertenece el actor. Este valor se encuentra predefinido en la tabla organización. El código permitirá mantener las diferencias entre las organizaciones que intervienen en el proyecto.
5. Autor: Código del autor que lleva a cabo el llenado de las tablas correspondientes.
6. Fecha: Fecha en la que se lleva a cabo la toma de información.
7. Tipo: Puede ser principal o secundario. El tipo se encuentra asociado con el nivel de complejidad y funcionalidad del producto.
8. Estado: Puede ser pendiente o concluido.
9. Comentario: Aquí se añade cualquier comentario asociado con el actor que pueda enriquecer o esclarecer su participación. Se debe anotar que la participación de una persona puede asumir diversos roles.

Atributos	Descripción
Código del actor	ACT-DDDD
Versión	VDD.DD
Rol	
Organización	
Autor	
Fecha	
Tipo	
Estado	
Comentario	

Tabla 3.3: Plantilla para actores

3.0.3 Autores

Los autores son las personas encargadas de tomar la información de las fuentes y de los actores para posteriormente plasmarla en las plantillas de educación, ilación y especificación. Los autores pertenecen a la organización que desarrolla el producto de software. La tabla 3.4 se refiere al diseño y cuyos campos tienen el siguiente significado:

1. Código del autor: Código compuesto por las letras AUT seguido de cuatro dígitos. Por ejemplo, AUT-0012.
2. Nombres y apellidos: Nombres, apellido paterno, apellido materno del autor. Se debe de escribir información completa para evitar confusiones.
3. Fecha: Fecha en la que se lleva a cabo el llenado de la tabla.
4. Rol: Rol que cumple el autor dentro de la organización, este puede ser: analista, diseñador, evaluador de software, analista, arquitecto de software, entre otros. Un autor puede tener varios roles los mismos que se redactan de manera independiente.
5. Organización: Código de la organización a la que pertenece el autor. Por ejemplo, ORG-0017.
6. Autor: Código del autor que hace el llenado de la tabla.
7. Estado: Puede ser pendiente o concluido. Pendiente cuando aún no se cuenta con la información completa y concluido en el caso contrario.
8. Comentario: Se redacta cualquier observación asociada con el autor con la finalidad de aclarar su intervención en el proyecto del software.

Atributos	Descripción
Código del autor	AUT-DDDD
Nombres y Apellidos	
Fecha	
Rol	
Organización	
Autor	
Estado	
Comentario	

Tabla 3.4: Plantilla para autores

3.0.4 Fuentes

La plantilla diseñada para este fin se muestra en la tabla 3.5 y cuyos campos presentan el siguiente significado:

1. Código de la fuente: Código de la fuente que se encuentra definido por las letras FUE seguido de cuatro dígitos. Por ejemplo: FUE-0045.
2. Versión: Versión de la tabla, consta de dos dígitos seguido por un punto y finalizando en dos dígitos; estos documentos pueden ser modificados por las organizaciones, pero no pueden ser modificados por los integrantes de la organización que construye el software. Por ejemplo, V1.12.
3. Nombre: Se especifica el nombre de la fuente, estos pueden ser documentos base, oficios, planos entre otros. Por ejemplo: Manual de Organización y Funciones (MOF).

4. Fecha fuente: Fecha de la entrevista o lectura de la fuente.
5. Fecha plantilla: Fecha del llenado de la plantilla con la información proporcionada por la fuente.
6. Autor: Código del autor que llena la tabla.
7. Estado: Puede ser pendiente o concluido
8. Comentario: Se redacta cualquier observación asociada con la fuente produciendo una mayor claridad en la utilidad del mencionado artefacto.

Atributos	Descripción
Código de la fuente	FUE-DDDD
Versión	VDD.DD
Nombre	
Fecha fuente	
Fecha plantilla	
Autor	
Estado	
Comentario	

Tabla 3.5: Plantilla para fuentes

3.0.5 Educación

La educación es la primera etapa de la toma de requisitos. Es la etapa donde el cliente vierte sus apreciaciones y estas son captadas por el analista. La tabla 3.6 muestra los campos cuya descripción es la siguiente:

1. Código educación: Cada tabla de educación presenta un código que permite identificar a la correspondiente educación. Este campo comienza con las letras EDU seguido de un guion y a continuación un número de cuatro dígitos en forma correlativa. Ejemplo: EDU-0001 que significa la primera educación.
2. Nombre: Nombre de la tabla de educación.
3. Versión: Comúnmente implica un refinamiento sucesivo de la necesidad del cliente hasta lograr concretizar la verdadera necesidad. Consta de la letra "v" seguido de uno o dos dígitos, un punto y finalmente dos dígitos. Ejemplo: v1.0, v1.03, v21.37.
4. Fecha. Fecha de llenado de la tabla de educación.
5. Autor: Se plasma el código del autor que lleva a cabo la fase de educación. Esta información es de vital importancia ya que si se comete un error se puede determinar quién es el analista que formula la educación.
6. Fuente: Se escribe el código de la fuente de donde se extrae la información. También puede referirse a un otro tipo de documentos pero que son los insumos desde donde se extrae el conocimiento.

7. Código de ilación: Código de las tablas de ilación asociadas con la tabla de educación.
8. Descripción: Relato, en forma escrita y clara, de la necesidad del cliente. En ello se pueden especificar las condicionalidades de la educación; es decir las precondiciones y las postcondiciones.
9. Importancia: Se describe la importancia de la educación pudiendo ser vital (es de suma importancia tratarlo de manera clara), medio (no requiere demasiada atención ya que se detecta que es un subproducto de otro requisito), baja (puede esperar un buen tiempo para ser desarrollado).
10. Estado: Es la situación de la educación pudiendo obtener la situación de pendiente o concluido. El pendiente implica que la educación se encuentra en estado de generación y el concluido significa que la educación ha pasado por el control de versiones y que es la versión final.
11. Comentario: Una descripción de detalles complementarios al elemento en educación o anotaciones que pueden servir para formular los requisitos en la siguiente etapa.

Atributos	Descripción
Código educación	EDU-DDDD
Nombre	Nombre de la tabla.
Versión	VDD.DD
Fecha	
Autor	AUT-DDDD
Fuente	FUE-DDDD
Código de ilación	ILA-DDDD
Descripción	
Importancia	
Estado	
Comentario	

Tabla 3.6: Plantilla de educación

3.0.6 Ilación

La ilación es la etapa donde el analista plasma su punto de vista con respecto al entendimiento del modelo del negocio de acuerdo a la información entregada por el cliente en la fase de educación. La tabla 3.7 muestra la plantilla cuya descripción es la siguiente:

1. Código: Cada tabla de ilación presenta un código que permite identificar a la correspondiente ilación. Este campo comienza con las letras ILA seguido de un guion y a continuación un número de cuatro dígitos en forma correlativa. Ejemplo: ILA-0001 que significa la tabla de ilación número 1.

2. Nombre: Nombre de la tabla de ilación, el mismo que debe guardar correspondencia con el objetivo del requisito.
3. Versión: Versión que comúnmente implica un refinamiento sucesivo de la necesidad del analista para entender el problema. Consta de la letra "v" seguido de uno o dos dígitos, un punto y finalmente dos dígitos. Ejemplo: v1.0, v1.03, v21.37.
4. Fecha: Fecha de llenado de la tabla de ilación.
5. Autor: Código del analista que lleva a cabo la tarea de entender el modelo del negocio.
6. Actor: Código del actor asociado con la tabla de ilación. Se debe comprender que el actor es un elemento clave dentro de la funcionalidad del sistema.
7. Fuente: Código de la fuente de donde se extrae el conocimiento para elaborar la tabla de ilación.
8. Código de educación: Código de las tablas de educación asociadas con la tabla de ilación.
9. Código de especificación: Código de las tablas de especificación asociadas con la tabla de ilación. Estos códigos son considerados claves para lograr la correspondiente trazabilidad.
10. Precondición: Especificación de las precondiciones necesarias.
11. Procedimiento: Es un relato, en forma escrita, de la forma como es entendido el modelo del negocio por parte del analista pudiendo incluir precondiciones y postcondiciones correspondientes.
12. Postcondición: Especificación de las postcondiciones.
13. Prioridad: Depende de la relación que guarde la información con el modelo del negocio y de la necesidad de implementación pudiendo ser alta, media o baja.
14. Código de artefactos asociados: Código de artefactos como diagrama de clases, secuencias y otros asociados con la tabla de ilación.
15. Importancia: Puede ser vital u opcional. Vital cuando la información es importante para el proceso y opcional cuando se trata del caso contrario.
16. Estado: Puede ser pendiente o concluido. Pendiente cuando aún no se tiene concluido el proceso de ilación con respecto al proceso de educación correspondiente.
17. Comentarios: Descripción de aquellos detalles complementarios al elemento en ilación y que dan orientación a la construcción del producto de software correspondiente.

Atributos	Descripción
Código	ILA-DDDD
Nombre	Nombre de la tabla de ilación.
Versión	VDD.DD
Fecha	
Autor	AUT-DDDD
Actor	ACT-DDDD

Fuente	FUE-DDDD
Código de educación	EDU-DDDD
Código de especificación	ESP-DDDD
Precondición	
Procedimiento	
Postcondición	
Prioridad	
Código de artefactos asociados	
Importancia	
Estado	
Comentario	

Tabla 3.7: Plantilla de ilación

3.0.7 Especificación

La especificación es la etapa en donde el analista transforma sus criterios deducidos del modelo del negocio en la etapa de ilación hacia descripciones entendibles por los codificadores. La tabla 3.8 muestra la plantilla de especificación; se muestra a continuación su descripción:

1. Código: Cada tabla de especificación contiene un código que identifica a la correspondiente especificación. Este campo comienza con las letras ESP seguido de un guión y a continuación un número de cuatro dígitos en forma correlativa. Ejemplo: ESP-0041.
2. Nombre: Nombre de la tabla de especificación.
3. Versión: Versión que comúnmente implica un refinamiento sucesivo de la necesidad del analista para entender el problema. Consta de la letra "v" seguido de uno o dos dígitos, un punto y finalmente dos dígitos. Ejemplo: v2.0, v1.15, v17.23.
4. Fecha: Fecha de llenado de la tabla de especificación.
5. Autor: Código del analista que lleva a cabo la tarea de entender el modelo del negocio.
6. Actor: Código del autor que elabora la tabla de especificación.
7. Fuente: Código de la fuente de donde se extrae la información para elaborar la tabla de especificación.
8. Código de ilación: Código de las tablas de ilación asociadas con la tabla de especificación.
9. Precondición: Especificación de las precondiciones necesarias.
10. Procedimiento: Es un relato, en forma escrita, de la forma como es entendido el modelo de negocio por parte del analista pudiendo incluir precondiciones y postcondiciones correspondientes.

11. Postcondición: Especificación de las postcondiciones.
12. Código de artefactos asociados: Código de artefactos como diagramas o interfaces gráficas de usuario asociados a la tabla de especificación.
13. Importancia: Puede ser vital u opcional.
14. Estado: Puede ser pendiente o concluido.
15. Comentario: Una descripción de aquellos detalles complementarios al elemento en especificación y que dan orientación a la construcción del producto de software.

Atributos	Descripción
Código de especificación	ESP-DDDD
Nombre	Nombre de la tabla de especificación.
Versión	VDD.DD
Fecha	
Autor	AUT-DDDD
Actor	ACT-DDDD
Fuente	FUE-DDDD
Código de ilación	ILA-DDDD
Precondición	
Procedimiento	
Postcondición	
Código de artefactos asociados	
Importancia	
Estado	
Comentario	

Tabla 3.8: Plantilla de especificación

3.0.8 Trazabilidad

La trazabilidad consiste en mantener una relación entre las tablas de educación, ilación y especificación de tal manera que cuando se detecte un error, inconsistencia, ambigüedad que sugiera algún cambio en el requisito, este sea detectable fácilmente y su corrección pueda alcanzar las tres etapas previstas en el catálogo de requisitos, permitiendo alcanzar un catálogo de requisitos consistente.

Si en el catálogo de requisitos se prevé una codificación para las interfaces gráficas de usuario y estos son introducidos en la etapa de especificación (tablas de especificación), entonces se puede detectar el efecto de los errores en el correspondiente interfaz. Lo mismo puede suceder si se codifican los diagramas que proporcionan los lenguajes de modelado,

por ejemplo, UML, y esta codificación es insertada en las tablas de ilación o especificación.

Esto sugiere que las plantillas propuestas pueden ser modificadas dependiendo de las necesidades sugeridas por el problema a resolver. Cada artefacto debe ser codificado y su representación insertada en las plantillas propuestas haciendo que se pueda establecer una trazabilidad adecuada. Un ejemplo de esta modificación se muestra en la tabla 3.9.

Atributos	Descripción
Código	ESP-DDDD
Nombre	
Versión	VDD.DD
Autor	AUT-DDDD
Fuente	FUE-DDDD
Tipo	
Descripción	
Prioridad	
Dependencia	ILA-DDDD, DC-0014, DS-0023, IU-0007
Comentarios	

Tabla 3.9: Plantilla de especificación modificada

Por ejemplo, si elaboramos un Diagrama de Clases podemos codificarla como DC-0014 así como un Diagrama de Secuencias (DS-0023) y un Interfaz de Usuario (IU-0007) entonces la plantilla de especificación puede ser modificado tal como se muestra en la tabla 3.9. Cuando se realice la trazabilidad entonces se podrá detectar y corregir, con mayor detalle, los efectos del error en cada uno de los artefactos comprometidos. También se puede modificar la estructura de la tabla de especificación como la mostrada en la tabla 3.10.

Atributos	Descripción
Código	ESP-DDDD
Nombre	
Versión	VDD.DD
Autor	AUT-DDDD
Fuente	FUE-DDDD
Tipo	
Descripción	
Prioridad	
Dependencia	ILA-DDDD
Otras dependencias	DC-0014, DS-0023, IU-0007
Comentarios	

Tabla 3.10: Modificación de la estructura de la plantilla de especificación

3.6 Procedimiento para elaborar el catálogo de requisitos

El procedimiento para elaborar el catálogo de requisitos es el siguiente:

1. Planificación del trabajo por parte del jefe del proyecto al que incluye entrevistas, elaboración de artefactos o definición de artefactos, selección de analistas. Esta actividad incluye tiempos y un plan de contingencia que permita contrarrestar los riesgos asociados a esta actividad.
2. Entrevistas planificadas con el cliente y de manera recursiva de tal forma que al final se logre elaborar el documento de educación de requisitos. La finalidad es que el analista comience a entender el modelo de negocio que se va a enfrentar y consolidar criterios adecuados para la construcción del producto de software.
3. Proceso de elaboración del documento de educación.
4. Proceso de elaboración del documento de ilación de requisitos por medio de la retroalimentación que lleva a cabo con el cliente. Se realizan tantas reuniones de trabajo hasta comprender el modelo del negocio. Este proceso lo hace el analista del sistema juntamente con el cliente.
5. Proceso de elaboración del documento de especificación de requisitos. Esta actividad se lleva a cabo como una retroalimentación de los procesos de educación e ilación por parte del analista. La retroalimentación es importante ya que permite hacer las correcciones del caso y en cascada correspondiendo al concepto de trazabilidad.
6. Proceso de toma de decisiones para incluir artefactos necesarios para la codificación del producto de software como, por ejemplo: diagramas de clase, diagramas de secuencia, diagramas de objetos, interfaces gráficas de usuario entre otros. Cada artefacto es codificado de forma que pueda ser incluido en las tablas de educación, ilación y especificación.
7. Hacer las consultas con el cliente con respecto al modelo del negocio entendido. Si se encuentran errores, inconsistencias o ambigüedades llevar a cabo la trazabilidad para establecer los lugares donde se debe de llevar a cabo el mantenimiento respectivo afectando solo a aquellas tablas asociadas a los errores encontrados.

3.7 Ejemplo de catálogo de requisitos

Este primer ejemplo se toma del producto de software construido para la empresa desarrolladora de software denominada SOFTWARE SOLUTIONS SAC, donde las plantillas fueron alteradas dependiendo de la complejidad del problema a tratar. Para el cliente pocas son las herramientas que hablan sobre la especificación de los requisitos de software, pero si hablan sobre un producto de trazabilidad, el stakeholder no entiende como se realiza este proceso pues al no contar con las etapas de educación, ilación y especificación de requisitos, no pueden relacionarse las actividades del cliente, analista y desarrollador. Lo que busca es una herramienta que contenga estos tres puntos para la primera etapa de desarrollo.

Los objetivos propuestos por el cliente para el desarrollo del producto solicitado son los siguientes:

1. Desarrollar una herramienta que contenga las etapas de educación, elicitation y especificación de requisitos para la primera etapa del desarrollo de software.
2. Generar un artefacto para cada proceso y de forma automatizada dentro de la herramienta, así como el control de versiones de cada requisito.
3. Gestionar los actores y la organización de ambos lados del proyecto, tanto para la organización cliente, como para la organización desarrolladora.

Para el proceso de educación de requisitos se hizo una entrevista al cliente quien entregó los primeros alcances sobre el software, que se desea desarrollar. La entrevista despeja varias dudas que se tenía y ayudó a tener una perspectiva más clara acerca del proyecto. A continuación, se presentan los puntos relevantes de la entrevista:

1. Dentro del desarrollo de algún requisito, el usuario es único para evitar los errores que puedan ocurrir a la hora de modificar algún requisito ya que puede existir confusión.
2. El sistema debe generar tres documentos indispensables, un documento para la educación, elicitation y especificación; es decir los artefactos para el análisis del proyecto desarrollado.
3. El sistema debe producir la exportación de estos artefactos en archivos con extensiones .doc y .pdf.
4. El sistema de gestión de requisitos recibe el nombre de REMAN, además está orientado a analistas o desarrolladores, es decir personas conocedoras de la gestión de requisitos.
5. El proyecto se desarrolla para escritorio, la herramienta tiene una estimación de dos años para su construcción.
6. El cliente proporciona algunas plantillas, las cuales se deben completar con las faltantes (requerimientos no funcionales).
7. Una educación contiene los criterios del cliente expresados bajo un criterio particular el mismo que debe ser interpretado por el analista.
8. Una ilación tiene una o varias educaciones. Bajo ningún criterio una ilación no puede tener relación con alguna tabla de educación.
9. Una especificación contiene una o varias ilaciones. No pueden existir tablas de especificación que no guarden relación con alguna tabla de ilación.
10. El documento histórico registra la versión, fecha y la razón del cambio para luego, en la trazabilidad, detectar lo bueno y lo malo o lo perdido.
11. El producto lleva la denominación de REMAN que es un acrónimo que no guarda relación con el nombre o filosofía de la gestión de requisitos.

Se empieza definiendo las tablas para la organización, actores, autores y fuentes con información extraída del cliente. Las tablas 3.11 a la tabla 3.20 muestran los resultados.

Atributos	Descripción
Código	ORG-0001
Versión	V1.0
Fecha	02/11/2015
Nombre	SOFTWARE SOLUTIONS SAC
Dirección	Urbanización Alto de la Luna L10 JLBR
Teléfono	963759143
Representante	Manuel Mayoría Salas
Contacto	Christian Incalla Nina
Condición	Solicitante

Tabla 3.11: Plantilla para la organización SOFTWARE SOLUTIONS SAC

Atributos	Descripción
Código	ORG-0002
Versión	V1.0
Fecha	02/11/2015
Nombre	Objetive Software
Dirección	Avenida Alfonso Ugarte 210 Cercado Arequipa
Teléfono	959177991
Representante	Oscar Montesinos Benavides
Contacto	Eliana Montesinos Roque
Condición	Empresa desarrolladora

Tabla 3.12: Plantilla para la organización Objetive Software

Atributos	Descripción
Código del actor	ACT-0001
Apellidos y Nombres	Manuel Mayoría Salas
Código de la organización	ORG-0001
Versión	V1.0
Rol	Gerente General
Comentario	Representa a la empresa solicitante y es el encargado de entregar la información del producto a construir.

Tabla 3.13: Plantilla para el actor 0001

Atributos	Descripción
Código del autor	AUT-0001
Apellidos y Nombres	José Vargas Huamán
Código de la organización	ORG-0002
Rol	Jefe de proyecto
Versión	V1.0

Comentario	Jefe de proyecto de la casa desarrolladora, también tiene la posibilidad de cumplir el rol de analista.
------------	---

Tabla 3.14: Plantilla para el autor 0001

Atributos	Descripción
Código del autor	AUT-0002
Apellidos y Nombres	Christian Incalla Nina
Código de la organización	ORG-0002
Rol	Analista
Versión	V1.0
Comentario	Analista encargado de elaborar el catálogo de requisitos.

Tabla 3.15: Plantilla para el autor 0002

Atributos	Descripción
Código de la fuente	FUE-0001
Nombre de la fuente	Manuel Mayoria Salas
Versión	V1.0
Comentario	Autor que proporciona la información en forma verbal o escrita.

Tabla 3.16: Plantilla para la fuente 0001

Atributos	Descripción
Código de la fuente	FUE-0002
Nombre de la fuente	Plantilla de educación
Versión	V1.0
Comentario	Plantilla para recabar la información en la etapa de educación.

Tabla 3.17: Plantilla para la fuente 0002

Atributos	Descripción
Código de la fuente	FUE-0003
Nombre de la fuente	Plantilla de ilación
Versión	V1.0
Comentario	Plantilla para recabar la información en la etapa de ilación.

Tabla 3.18: Plantilla para la fuente 0003

Atributos	Descripción
Código de la fuente	FUE-0004

Nombre de la fuente	Plantilla de especificación
Versión	V1.0
Comentario	Plantilla para recabar la información en la etapa de especificación.

Tabla 3.19: Plantilla para la fuente 0004

Atributos	Descripción
Código de la fuente	FUE-0005
Nombre de la fuente	Catálogos de muestra
Versión	V1.0
Comentario	Catálogos de experiencias anteriores que sirven de muestra para la automatización.

Tabla 3.20: Plantilla para la fuente 0005

Seguidamente se procede a especificar los cuadros de educación, las mismas que se exponen desde la tabla 3.21 hasta la tabla 3.24. Los de ilación que se muestran desde la tabla 3.25 hasta la tabla 3.36 y los de especificación se muestran desde la tabla 3.37 hasta la tabla 3.48.

Atributos	Descripción
Código	EDU-0001
Nombre	Gestión del sistema
Versión	V2.1
Autor	AUT-0001
Fuente	FUE-0001
Descripción	El cliente solicita el acceso al programa principal. Lo hace presionando el ícono correspondiente.
Importancia	Vital
Estado	Concluido
Comentario	Ninguno

Tabla 3.21: Plantilla de educación 0001

Identificadas las primeras plantillas entonces se procede a definir e implementar las siguientes plantillas, es decir, las plantillas de educación, ilación y especificación. Estas deben ser llenadas con los criterios del analista y retroalimentadas dependiendo del tipo de problema encontrado; es decir inconsistencias y/o ambigüedades.

Atributos	Descripción
Código	EDU-0002
Nombre	Gestión de requisitos
Versión	V2.1
Autor	AUT-0002
Fuente	FUE-0001, FUE-0002, FUE-0003, FUE-0004, FUE-0005

Descripción	El cliente solicita la construcción de un gestor de requisitos, que genera inicialmente tres libros correspondientes a las tres etapas de la Gestión de Requisitos. Estos requisitos deben ser clasificados en requisitos funcionales y no funcionales.
Importancia	Vital
Estado	Concluido
Comentario	El sistema de gestión de requisitos está dirigido al personal calificado en la gestión de requisitos, contando con alguna experiencia en herramientas de gestión de requisitos como REM (Requirement Management) y otros que se encuentren en el mercado (ejemplo: Rational Requisite Pro, CaliberRM, Integral Requisite Analyzer, DRES, OSRMT, DOORS, Visure Requirements, Reqify, Jama entre otros).

Tabla 3.22: Plantilla de educación 0002

Atributos	Descripción
Código	EDU-0003
Nombre	Gestión de datos generales
Versión	V2.1
Autor	AUT-0002
Fuente	FUE-0001, FUE-0005
Descripción	El cliente solicita la gestión de datos generales dentro del desarrollo de un software. Estos datos pertenecen a la empresa solicitante, empresa desarrolladora, autores y actores. Esta información debe ser codificada con el objetivo de mantener una relación con los requisitos respectivos.
Importancia	Vital
Estado	Concluido
Comentario	Esta información es considerada como persistente.

Tabla 3.23: Plantilla de educación 0003

Atributos	Descripción
Código	EDU-0004
Nombre	Gestión de libros
Versión	V2.1
Autor	AUT-0002
Fuente	FUE-0001, FUE-0005
Descripción	El cliente solicita que en la herramienta se defina el concepto de libros con el nombre de las etapas solicitadas. Asimismo, se debe de insertar el concepto de versiones en las tablas.
Importancia	Vital
Estado	Concluido
Comentario	Esta información es considerada como persistente durante el ciclo de desarrollo del producto de software y en la herramienta las etapas deben definirse como libros.

Tabla 3.24: Plantilla de educación 0004

Luego, se pasa a elaborar las tablas de ilación las mismas que se muestran desde la tabla 3.25 hasta la tabla 3.30.

Atributos	Descripción
Código	ILA-0001
Nombre	Acceso al programa principal
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	El acceso al programa principal será libre para cualquier persona que tenga acceso al computador o laptop donde se encuentre instalado el software, para ello debe tener previsto un usuario y una contraseña.
Prioridad	Alta
Dependencia	EDU-0001
Datos específicos	DNI del usuario
Comentarios	No se precisa si se pueden compartir proyectos entre diferentes maquinas o varios en una misma máquina, pero el cliente solicita prever el control de concurrencia para el mantenimiento de la herramienta construida. El objetivo de ello es prever la ampliación de la herramienta en un futuro próximo lo que implica su uso como un sistema distribuido o distribuido en tiempo real. El usuario inicia la sesión con su usuario y contraseña referido al número de DNI. Posteriormente puede ser cambiada la contraseña por la misma persona.

Tabla 3.25: Plantilla de ilación 0001

Atributos	Descripción
Código	ILA-0002
Nombre	Gestión del proyecto a desarrollar
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá elegir un proyecto, dentro de la lista de proyectos; una vez seleccionado el proyecto, se cargarán todos los datos de los libros correspondientes, además se presentan opciones como cambiar de proyecto actual el cual consiste en el cierre del proyecto actual (o proyecto que está empleando el usuario) para luego regresar a la pantalla inicial con la lista de proyectos, pudiendo elegir otro proyecto el mismo que alberga los tres libros definidos.</p> <p>Precondición: haber creado un proyecto o el proyecto escogido debe contener la información correspondiente.</p> <p>Postcondición: el usuario podrá realizar nuevos ingresos, cambios, eliminación y control de versiones de cada elemento del libro; esto significa un refrescamiento o actualización de datos.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> 1. El usuario luego de ingresar al programa se encuentra con proyectos creados para su selección. 2. El usuario ingresa al proyecto seleccionado. 3. El sistema carga todos los datos del proyecto, es decir los libros con la información de la educación, ilación y especificación. 4. El sistema le permite al usuario salir del proyecto actual para poder ingresar y editar otro. 5. El usuario podrá modificar la información del proyecto. 6. El usuario podrá eliminar el proyecto el mismo que pierde toda la información. <p>La excepción se presenta cuando ningún proyecto ha sido creado.</p>
Prioridad	Alta
Dependencia	EDU-0001

Datos específicos	Nombre del proyecto
Comentarios	La carga de los datos de un proyecto se realiza al hacer la consulta del archivo XML.

Tabla 3.26: Plantilla de ilación 0002

Atributos	Descripción
Código	ILA-0003
Nombre	Educación de requisitos
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La educación de requisitos es una sección del sistema, en ella se registrarán los requisitos a educacionar, contando con un sistema de versiones, así como también de actores y especialistas que intervienen en la educación; además cuenta con las opciones de modificar, eliminar y versionar.</p> <p>Precondición: el usuario debe estar ejecutando el programa principal. Deben de estar creados los especialistas que intervienen en la educación. Deben de estar creadas las fuentes que intervienen en la educación.</p> <p>Postcondición: el sistema almacena la reducción de requisitos.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de agregar una nueva educación. 2. El usuario ingresa de manera obligatoria los siguientes campos: nombre del requisito a educacionar, número del requisito a educacionar, versión de la educación, datos de la fuente (nombre, cargo), datos del autor (nombre, especialidad, tipo y experiencia), datos de la educación (tipo, objetivo y fecha) y descripción de la educación. 3. El usuario puede ingresar de manera no obligatoria el campo: observaciones. 4. El usuario confirma la creación de una nueva educación. 5. El usuario puede modificar el requisito. 6. El usuario puede eliminar el requisito.
Prioridad	Alta
Dependencia	EDU-0004
Datos específicos	Información solicitada en la descripción.
Comentarios	Se pueden generar procedimientos de seguridad para el control de creación y modificación de educciones. Cada educación de requisitos se almacena en archivos XML junto con las demás educciones creadas. La carga de los datos de un proyecto se realiza al hacer la consulta del archivo XML.

Tabla 3.27: Plantilla de ilación 0003

Atributos	Descripción
Código	ILA-0004
Nombre	Ilación de requisitos
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional

Descripción	<p>La ilación de requisitos es una sección del sistema, en ella se registra los requisitos del sistema orientados al analista tomando en cuenta que una ilación puede contener una o más educciones, el sistema de versiones, así como contar con las opciones de modificar, eliminar y versionar.</p> <p>Precondiciones: debe estar creada con anterioridad la sección de educación, los actores que intervienen en la ilación y las fuentes que intervienen en la ilación.</p> <p>Secuencia normal:</p> <ul style="list-style-type: none"> • El usuario selecciona la opción de agregar una nueva ilación. • El usuario ingresa de manera obligatoria los siguientes campos: nombre de la ilación, número de la ilación, código de la educación relacionada, versión de la ilación, fecha cuando se lleva a cabo la ilación, autor o autores, fuentes, dependencias, descripción. • Precondición, secuencia normal. • Postcondición, excepciones. • Secuencia normal, Postcondición y excepciones. • El usuario puede ingresar de manera no obligatoria el campo comentarios. • El usuario puede modificar el requisito. • El usuario puede eliminar el requisito.
Prioridad	Alta
Dependencia	EDU-0004
Datos específicos	Información solicitada en la descripción.
Comentarios	<p>Se pueden generar procedimientos de seguridad para el control de la creación y modificación de la ilación y poder percibir si faltan los correspondientes archivos.</p> <p>Cada ilación de requisito se almacena en archivos XML junto con las demás ilaciones creadas.</p>

Tabla 3.28: Plantilla de ilación 0004

Atributos	Descripción
Código	ILA-0005
Nombre	Especificación de requisitos
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La especificación de requisitos es una sección del sistema, en ella se registran los requisitos del sistema orientados al desarrollador, tomando en cuenta que una especificación puede contener una o más ilaciones, además del sistema de versiones, así como las opciones de modificar, eliminar y versionar.</p> <p>Precondiciones: debe estar creada con anterioridad la ilación relacionada, los actores que intervienen en la especificación de requisitos, las fuentes que intervienen en la especificación de requisitos.</p> <p>Postcondición: el sistema guardará la especificación del requisito en el proyecto.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de agregar una nueva especificación de requisito. 2. El usuario ingresa de manera obligatoria los siguientes campos: nombre de la especificación de requisito de software, número de la especificación de requisito a especificar, código de la ilación relacionada, versión, fecha de la especificación, autor(es), fuentes, dependencias, descripción, precondición, postcondición y excepciones. 3. El usuario puede ingresar de manera no obligatoria el campo comentarios. 4. El usuario podrá modificar el requisito. 5. El usuario podrá eliminar el requisito.
Prioridad	Alta
Dependencia	EDU-0004
Datos específicos	Información solicitada en la descripción.

Comentarios	<p>Se pueden generar procedimientos de seguridad para el control de creación y modificación de especificación.</p> <p>Cada especificación de requisito se almacenará en archivos XML junto con las demás especificaciones creadas.</p>
-------------	--

Tabla 3.29: Plantilla de ilación 0005

Atributos	Descripción
Código	ILA-0006
Nombre	Requisitos funcionales
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>Los requisitos funcionales es una sección del sistema, en ella se registrarán los requisitos del sistema orientados al rendimiento, y aspectos de calidad, además de contar con las opciones de modificar, eliminar y versionar.</p> <p>Los requisitos funcionales deben estar claramente definidos para lograr su clasificación adecuada.</p> <p>Precondición: el usuario debe estar en el programa principal o el programa principal debe estar ejecutándose.</p> <p>Postcondición: el sistema guardará el estado del requisito en el proyecto.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El usuario selecciona la opción de agregar un nuevo requisito funcional. El usuario ingresa de manera obligatoria los siguientes campos: nombre del requisito funcional, identificador del requisito funcional, descripción del requisito funcional, autor del requisito funcional. El sistema completará el campo cargo del autor. El usuario podrá modificar un requisito funcional. El usuario podrá eliminar un requisito funcional.
Prioridad	Alta
Dependencia	EDU-0002
Datos específicos	Información solicitada en la descripción: nombre del requisito funcional, identificador del requisito funcional, descripción del requisito funcional.
Comentarios	<p>El ingreso del requisito funcional es tal como lo entiende el analista y posteriormente, producto del refinamiento, se puede almacenar en las plantillas orientadas para esta finalidad.</p> <p>Los requisitos funcionales son el complemento de los requisitos no funcionales presentando el mismo grado de importancia para el analista.</p>

Tabla 3.30: Plantilla de ilación 0006

Atributos	Descripción
Código	ILA-0007
Nombre	Gestión de actores fuente
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional

Descripción	<p>La gestión de actores fuente es una sección del sistema, en ella se registrará la información de los actores de la organización que solicita el sistema.</p> <p>Además, podrá contar con las opciones de modificar y eliminar la información del mencionado actor.</p> <p>Precondición: la organización a la cual pertenece el actor debe estar registrada en el sistema. Su registro debe contemplar toda la información relacionada con la organización.</p> <p>Postcondición: el sistema guardará el estado del actor fuente en el proyecto.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El usuario agregará un nuevo actor. El usuario ingresa de manera obligatoria los siguientes campos: número del actor, datos del actor, la organización a la que pertenece, el cargo que ocupa, el tipo (desarrollador, cliente, usuario final), correo electrónico, comentarios. El usuario podrá modificar un actor. El usuario podrá eliminar un actor.
Prioridad	Alta
Dependencia	EDU-0003
Datos específicos	Información solicitada en la descripción: número del actor, datos del actor, la organización a la que pertenece, el cargo que ocupa, el tipo (desarrollador, cliente, usuario final), correo electrónico, comentarios.
Comentarios	Los actores fuente son objetos persistentes en el proyecto. La información de los mismos permanecerá en todo el medio así cambie de uso el libro. Esto se debe a que tienen relación con las etapas de educación, ilación y especificación de requisitos.

Tabla 3.31: Plantilla de ilación 0007

Atributos	Descripción
Código	ILA-0008
Nombre	Gestión de actores especialistas
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La gestión de actores especialistas es una sección del sistema, en ella se registrarán los actores especialistas de la organización que desarrolla el proyecto.</p> <p>Además, el sistema contará con las opciones de modificar y eliminar.</p> <p>Precondición: la organización a la cual pertenece el actor debe de estar registrada en el sistema.</p> <p>Postcondición: el sistema guardará el estado del actor especialista en el proyecto.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El usuario agregará un nuevo actor especialista. El usuario ingresa de manera obligatoria los siguientes campos: número del actor, datos del actor, la organización a la que pertenece, el cargo que ocupa, el tipo (desarrollador, cliente, usuario final), correo electrónico y comentarios. El usuario podrá modificar un actor especialista. El usuario podrá eliminar a un actor especialista.
Prioridad	Alta
Dependencia	EDU-0003
Datos específicos	Información solicitada en la descripción: número del actor, datos del actor, la organización a la que pertenece, el cargo que ocupa, el tipo (desarrollador, cliente, usuario final), correo electrónico y comentarios.
Comentarios	Los actores especialistas son objetos persistentes en el proyecto. Esto implica que su información permanece en todo el proceso de ejecución del aplicativo; y lo hace de manera indefinida en el disco duro del dispositivo.

Tabla 3.32: Plantilla de ilación 0008

Atributos	Descripción
Código	ILA-0009
Nombre	Gestión de la organización
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La gestión de organizaciones es una sección del sistema, en ella se registrará la información de las organizaciones tanto del cliente como de la casa desarrolladora de software.</p> <p>Precondición: el usuario debe estar ejecutando el programa principal.</p> <p>Postcondición: el sistema guardará el estado de la organización en el proyecto.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El usuario agregará una nueva organización producto de la correspondiente acción. El usuario podrá registrar la organización con los siguientes campos: número de organización, datos de la organización, dirección, teléfono, correo electrónico y comentarios. El usuario podrá modificar la información de una organización. El usuario podrá eliminar la información de una organización.
Prioridad	Alta
Dependencia	EDU-0003
Datos específicos	Información solicitada en la descripción: número de organización, datos de la organización, dirección, teléfono, correo electrónico y comentarios.
Comentarios	Las organizaciones son objetos persistentes en el proyecto. Su información será almacenada en el disco duro del dispositivo y será usado en todo el uso de la aplicación. La eliminación de la información de la organización implica una eliminación lógica más no una eliminación física. Se respeta la definición de persistencia.

Tabla 3.33: Plantilla de ilación 0009

Atributos	Descripción
Código	ILA-0010
Nombre	Gestión del proyecto
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La gestión de proyectos es una sección del sistema, en ella se registrarán los principales datos del proyecto en desarrollo.</p> <p>Además, el sistema podrá realizar las opciones de modificación y eliminación.</p> <p>Precondición: en el sistema deben estar registradas las organizaciones.</p> <p>Postcondición: el sistema guardará el estado del proyecto en el sistema.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El usuario agregará un nuevo proyecto. El usuario podrá registrar el proyecto con los siguientes campos: código del proyecto, nombre del proyecto, nombre del producto a desarrollar, nombre de la empresa desarrolladora, nombre de la empresa cliente, fecha de inicio del proyecto, fecha de finalización del proyecto, apellidos y nombres del líder del proyecto. El usuario podrá modificar el proyecto. El usuario podrá eliminar el proyecto.
Prioridad	Alta
Dependencia	EDU-0001

Datos específicos	Información solicitada en la descripción: código del proyecto, nombre del proyecto, nombre del producto a desarrollar, nombre de la empresa desarrolladora, nombre de la empresa cliente, fecha de inicio del proyecto, fecha de finalización del proyecto, apellidos y nombres del líder del proyecto.
Comentarios	Los proyectos son objetos persistentes en el proyecto. La eliminación de la información del proyecto implica una eliminación lógica más no una eliminación física. Se respeta la definición de persistencia.

Tabla 3.34: Plantilla de ilación 0010

Atributos	Descripción
Código	ILA-0011
Nombre	Gestión del historial de libros
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La gestión de versiones en el documento histórico es una sección del sistema, en ella se registrarán los cambios producidos dentro de la gestión de requisitos es decir la educación, ilación, especificación y los requisitos no funcionales.</p> <p>Las versiones de la educación, ilación y la especificación de requisitos forman una versión conjunta que se agrega a la denominación del libro.</p> <p>Además, el sistema ofrecerá las opciones de exportar el libro seleccionado y la versión deseada.</p> <p>Precondición: se debe de agregar o modificar un requisito.</p> <p>Postcondición: el sistema registrará y actualizará el historial de versiones en el dispositivo usado en ese momento.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El sistema actualizará el historial de versiones para cada libro con los siguientes datos: versión, motivos del cambio, autor y fecha.
Prioridad	Alta
Dependencia	EDU-0004
Datos específicos	Información solicitada en la descripción: versión, motivos del cambio, autor y fecha.
Comentarios	<p>Una secuencia se describirá de la siguiente manera, al entrar a la sección de historial del sistema, se presentarán los cuatro libros a modo de botón o como subcarpetas de una carpeta (ejemplo: como el explorador de archivos en el sistema operativo Windows).</p> <p>Luego al seleccionar un libro, en la pantalla continua (donde se trabaja), se presentarán en lista las versiones del libro y con detalles como el motivo de la versión o como también la fecha de la versión.</p>

Tabla 3.35: Plantilla de ilación 0011

Atributos	Descripción
Código	ILA-0012
Nombre	Gestión de reportes
Versión	V1.0
Autor	AUT-0002
Fuente	FUE-0001
Tipo	Funcional

Descripción	<p>La gestión de reportes es una sección del sistema, con ella se podrá generar un reporte resumen de todo el proyecto, con los datos que se seleccione, en el caso de ser libros, se escogerán los registrados en su última versión.</p> <p>Los reportes no son dinámicos ya que es el usuario quien escogerá el tipo de reporte.</p> <p>Precondición: el usuario se encuentra en el programa principal.</p> <p>Postcondición: el sistema generará el resumen con los campos solicitados.</p> <p>Secuencia normal:</p> <ol style="list-style-type: none"> El sistema permitirá elegir entre todas las opciones siguientes: proyecto, organización, libro de educciones, libro de ilaciones, libro de especificaciones, libro de requisitos no funcionales y comentarios. El usuario solicitará la generación del reporte necesitado. El sistema generará el reporte y desplegará las opciones de Mostrar (Vista Previa) y Exportar (Exportar reporte PDF para luego imprimirse).
Prioridad	Alta
Dependencia	EDU-0001, EDU-0002
Datos específicos	Información asociada con los libros de educación, ilación y especificación de requisitos.
Comentarios	<p>Una secuencia se describiría de la siguiente manera, al entrar a la sección de historial del sistema, se presentarán los cuatro libros a modo de botón o como subcarpetas de una carpeta (ejemplo, explorador de archivos en Windows).</p> <p>Luego al seleccionar un libro en la pantalla continua (donde se trabaja), se presentarán en lista las versiones del libro, con detalles como el motivo de la versión, como también de la fecha de la versión.</p>

Tabla 3.36: Plantilla de ilación 0012

En vista de que la cantidad de tablas para la fase de especificación de requisitos es extensa, solo se presentan algunas de ellas para lograr el objetivo y alcanzar, finalmente, la trazabilidad de requisitos.

Atributos	Descripción
Código	ESP-0001
Nombre	Ingresar al programa principal
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá acceder al programa principal si este se encuentra correctamente instalado.</p> <p>Precondición: el sistema debe estar correctamente instalado.</p> <p>Postcondición: el sistema muestra la interfaz principal siendo la opción, a primera vista, la selección de un proyecto para continuar con la edición o acumulación de información; en caso sea el primer uso el sistema mostrará una imagen a modo de bienvenida (imagen previamente definida por el equipo de desarrollo de interfaces gráficas de usuario) y sólo le quedará al usuario registrar el primer proyecto, como también las organizaciones y actores. Para ello se debe de subir la imagen con tamaño 1 cm x 1 cm.</p> <p>Excepciones: el sistema se encuentra mal instalado en el computador, o con archivos faltantes, lo que conlleva a no poder iniciar el sistema.</p>
Prioridad	Alta
Dependencia	ILA-0001

Comentarios	<p>Las consideraciones de cómo se muestre la lista de proyectos es libre decisión de diseño, al cliente no le interesa un orden predefinido por lo que el codificador será el encargado de mostrar la lista correspondiente bajo un criterio en particular.</p> <p>Solo tener en cuenta que no es necesario abrir algún proyecto, para ingresar una organización o actores, estos datos son generales, al momento de crear un proyecto se hace referencia a estos datos haciendo que el usuario tenga la posibilidad de hacer la elección del caso.</p>
-------------	---

Tabla 3.37: Plantilla de especificación de requisitos 0001

Atributos	Descripción
Código	ESP-0002
Nombre	Salir del programa principal
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá salir del programa principal desde la opción salir o cerrar del menú archivo, o al hacer clic en los botones correspondientes.</p> <p>Precondición: El usuario podrá salir del programa principal si este se encuentra correctamente activo. El sistema debe estar iniciado.</p> <p>Postcondición: el sistema cierra el programa principal, dando la opción de poder guardar los cambios realizados. Al guardar, todo cambio se guarda sobre la versión actual en la que se esté trabajando, caso contrario se pierde la información y luego se cierra la aplicación.</p> <p>Excepciones: el sistema se encuentra mal instalado en el computador, o con archivos faltantes por lo que no se puede trabajar con el aplicativo.</p>
Prioridad	Media
Dependencia	ILA-0001
Comentarios	<p>El cierre del aplicativo conlleva a que la información pueda o no permanecer en el contexto del mismo.</p> <p>La decisión del usuario es la que debe de prevalecer. Se debe de tener en consideración que si existe una falla o corte del fluido eléctrico la información es destruida inmediatamente salvo el caso previsto de contar con un UPS en la cual le otorga al usuario la potestad de cerrar el aplicativo.</p> <p>Al activarse el UPS el sistema debe enviar un mensaje solicitando la grabación de la información para luego de la decisión del usuario cerrarse automáticamente.</p> <p>En caso de que el aplicativo se cierre y el usuario lo considera una equivocación entonces el usuario puede volver a cargar el proyecto y el aplicativo reacciona ante esto de manera automática con toda la información relacionada con el proyecto.</p>

Tabla 3.38: Plantilla de especificación de requisitos 0002

Atributos	Descripción
Código	ESP-0003
Nombre	Ingresar a proyecto seleccionado
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario ingresa a un proyecto seleccionado entre la lista de proyectos existentes.</p> <p>Precondición: el sistema debe estar iniciado. En el sistema deben existir proyectos desarrollados.</p> <p>Postcondición: el sistema carga todos los datos del proyecto, los que incluyen a: Libro de educciones, libro de ilaciones, libro de especificaciones, libro de requerimientos no funcionales, datos generales del proyecto.</p>
Prioridad	Alta

Dependencia	ILA-0002
Comentarios	Este paso se presenta a la hora de iniciar el proyecto y si el usuario se encuentra en un proyecto, y desea acceder a otro, primero deberá cerrar el proyecto actual, luego el sistema regresa al estado inicial, es decir al estado en que puede escoger un proyecto de la lista de proyectos.

Tabla 3.39: Plantilla de especificación de requisitos 0003

Atributos	Descripción
Código	ESP-0004
Nombre	Salir del proyecto actual
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	El usuario podrá salir del proyecto actual, luego podrá seleccionar otro proyecto, pues se encontrará en el estado inicial (donde se muestra la lista de proyectos) o crear un proyecto nuevo. Precondición: el sistema debe estar iniciado y se debe estar dentro de un proyecto. Postcondición: el sistema regresa a la pantalla principal.
Prioridad	Alta
Dependencia	ILA-0002
Comentarios	Al salir del proyecto actual si no se han guardado los cambios, el sistema pedirá la confirmación para guardar los cambios de cada requerimiento editado.

Tabla 3.40: Plantilla de especificación de requisitos 0004

Atributos	Descripción
Código	ESP-0005
Nombre	Agregar educación de requisitos
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	La educación de requisitos es una sección del sistema, en ella se registrarán todos los datos que involucran a una educación como son: nombre del requisito a educacionar, datos del actor, fuente (nombre, cargo y tipo), datos del actor especialista (nombre, especialidad, experiencia, cargo), datos de la educación TIPO, datos de la educación OBJETIVO, descripción de la educación, observaciones del requisito educido, código del requisito a educacionar, versión de la educación, datos de la educación FECHA. Precondición: el usuario deberá estar ejecutando el programa principal. En el sistema deben de estar creados los especialistas que intervienen en la educación. En el sistema deben de estar creadas las fuentes que intervienen en la educación. El usuario debe de estar dentro de algún proyecto. El usuario debe de estar dentro del libro de EDUCACION o desde el menú de herramientas (opcional). Postcondición: el sistema guardará la educación del requisito en el proyecto.
Prioridad	Alta
Dependencia	ILA-0003

Comentarios	<p>Se implementará un código autoincremental para que en cada proyecto este detecte el código del último requisito dentro del libro correspondiente, por eso es automático, propio del sistema.</p> <p>El actor deberá aparecer en una lista desplegable si se encuentra registrado, la búsqueda por coincidencias ayudará en el caso de ser muchos los participantes.</p> <p>El usuario ingresará el tipo de educación, así como los objetivos de este, siendo concisos y claros.</p>
-------------	--

Tabla 3.41: Plantilla de especificación de requisitos 0005

Atributos	Descripción
Código	ESP-0006
Nombre	Modificar el requisito educacionado
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá modificar un requisito con la sola acción de un botón que le permitirá editar los campos de la plantilla, al finalizar el usuario podrá guardar los cambios en cuyo caso se guardará sobre la versión en que se está trabajando o versionar el requisito que permitirá al usuario ingresar una nueva versión y guardarla.</p> <p>La modificación implica el cambio de por lo menos un carácter en el texto; si el texto introducido supera los límites mostrados para el ingreso de la información, el aplicativo debe mantener una estricta comunicación con el sistema operativo del dispositivo para generar los espacios correspondientes.</p> <p>Precondición: el requisito modificado debe de existir, por lo menos en su versión primaria.</p> <p>Postcondición: el sistema guardará el requisito.</p>
Prioridad	Alta
Dependencia	ILA-0003
Comentarios	<p>La modificación permitirá editar los campos de cada requisito, bloqueando los datos del código del requisito, lo que implica que solo pueden ser modificados aquellos campos que presentan el permiso correspondiente; si los límites son superados entonces el sistema operativo los debe gestionar a partir de una petición del codificador insertado en el código.</p> <p>Si la educación requiere de la intervención del analista, este es el responsable de mantener la información con criterios de aceptación para los codificadores.</p> <p>El codificador no resuelve el problema, el cliente simplemente recibe la redacción del requisito para tomar las decisiones del caso.</p>

Tabla 3.42: Plantilla de especificación de requisitos 0006

Atributos	Descripción
Código	ESP-0012
Nombre	Agregar requisito ilado
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional

Descripción	<p>La ilación de requisitos es una sección del sistema, en ella se registrarán todos los datos que involucran a una ilación como son:</p> <ul style="list-style-type: none"> • Datos que el usuario llena o selecciona. • Nombre del requisito a elicitar. • Código del requisito educido al que hace referencia. (Selezionable) • Datos del actor fuente (nombre)(Selezionable de entre los registrados en la organización cliente definida en la creación del proyecto). • Datos del actor especialista (nombre). (Selezionable de entre los registrados en la organización desarrolladora definida en la creación del proyecto). • Dependencias (áreas con los que se relaciona)(Selezionable dentro de las áreas predeterminadas). • Datos de la educación TIPO. • Precondición (precondiciones para que funcione la ilación). • Descripción de la ilación.
Prioridad	Alta
Dependencia	ILA-0004
Comentarios	<p>Se implementará un código autoincremental para que en cada proyecto este detecte el código del último requisito dentro del libro correspondiente, por eso es automático, propio del sistema y para lo cual:</p> <p>El actor deberá aparecer en una lista desplegable si se encuentra registrado, la búsqueda por coincidencias ayudará en el caso de ser muchos los participantes.</p> <p>El usuario deberá de elegir entre las educciones existentes para establecer la relación con la ilación.</p> <p>El sistema es automático y detectará los requerimientos educacionados siendo necesario solo elegirlo.</p>

Tabla 3.43: Plantilla de especificación de requisitos 0012

Atributos	Descripción
Código	ESP-0013
Nombre	Modificar requisito ilado
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá modificar un requisito ilado con la sola acción de un botón que le permitirá editar los campos de la plantilla, al finalizar el usuario podrá guardar los cambios en cuyo caso se guardará sobre la versión en que se esté trabajando o versionar el requisito que permitirá al usuario ingresar una nueva versión y guardarla.</p> <p>El analista debe tener la certeza de que al modificar un requisito ilado, este guarda relación con los requisitos educacionados por lo que es posible que la corrección la deba de hacer en cascada.</p> <p>Precondición: el requisito a modificar debe de existir, por lo menos en su versión primaria.</p> <p>Postcondición: el sistema guardará el requisito.</p> <p>Excepciones: el número de versión debe ser superior al predecesor.</p>
Prioridad	Alta
Dependencia	ILA-0004
Comentarios	<p>La modificación permitirá editar los campos de cada requisito, bloqueando los datos del código del requisito.</p> <p>Se debe recordar que es responsabilidad del analista el guardar la concordancia con respecto a las tablas de educación por lo que deberá aparecer un mensaje en el sentido de indicar al analista que debe de hacer las modificaciones en los requisitos educacionados que guarden relación con los requisitos ilados.</p> <p>El control de versiones debe ser concordante con la parte de la ilación y la educación y las políticas de ellos deben ser fijadas por el jefe del proyecto.</p> <p>En la modificación, si existiera, obliga a cambiar de versión tanto en el proceso de ilación como en el proceso de educación.</p>

Tabla 3.44: Plantilla de especificación de requisitos 0013

Atributos	Descripción
Código	ESP-0014
Nombre	Agregar requisitos sobre especificación
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>La especificación de requisitos es una sección del sistema, en ella se registrarán todos los datos que involucran a una especificación como son: Datos que el usuario llena o selecciona:</p> <ul style="list-style-type: none"> • Nombre del requisito a especificar. • Código del requisito ilado al que hace referencia. • Datos del actor fuente (nombre). • Datos del actor especialista (nombre). • Dependencias (áreas con los que se relaciona). • Precondición (precondiciones para que funcione la especificación). • Descripción de la especificación. • Postcondición (postcondiciones de la ilación) • Excepciones • Comentarios (opcional). • Código del requisito a especificar. • Versión de la educación. • Datos de la especificación FECHA (fecha cuando se especifica el requisito). <p>Precondición: el usuario deberá estar ejecutando el programa principal. En el sistema deben de estar creados los especialistas que intervienen en la ilación. En el sistema deben de estar creadas las fuentes que intervienen en la ilación. El usuario debe de estar dentro de algún proyecto. El usuario debe de estar dentro del libro de ESPECIFICACIÓN o desde el menú de herramientas (opcional). El usuario debe haber registrado la ilación del requisito que se necesita.</p> <p>Postcondición: el sistema guardará la especificación del requisito en el proyecto.</p>
Prioridad	Alta
Dependencia	ILA-0005
Comentarios	<p>Se implementará un código autoincremental para cada proyecto este detecte el código del último requisito dentro del libro correspondiente, por eso es automático y propio del sistema.</p> <p>Se deben llenar los campos expuestos de manera obligatoria.</p>

Tabla 3.45: Plantilla de especificación de requisitos 0014

Atributos	Descripción
Código	ESP-0015
Nombre	Modificar un requisito especificado
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional

Descripción	<p>El usuario podrá modificar un requisito con la sola acción de un botón que le permitirá editar los campos de la plantilla, al finalizar el usuario podrá guardar los cambios en cuyo caso se guardará sobre la versión en que se esté trabajando o versionar el requisito que permitirá al usuario ingresar una nueva versión y guardarla. Por otro lado, el aplicativo debe lanzar una advertencia de que, si ha surgido un cambio en la especificación de requisitos, es probable que exista una modificación en las tablas de ilación y probablemente en las tablas de educación.</p> <p>Precondición: el requisito a modificar debe de existir, por lo menos en su versión primaria.</p> <p>Postcondición: el sistema guardará el requisito.</p> <p>Excepciones: si el requisito no existe enviar un mensaje de advertencia.</p>
Prioridad	Alta
Dependencia	ILA-0005
Comentarios	<p>La modificación permitirá editar los campos de cada requisito, bloqueando los datos del código del requisito.</p> <p>Tener en cuenta que si se modifica un requisito dentro de la especificación puede generar modificaciones en los requisitos ilados relacionados e incluso en los requisitos educacionados relacionados.</p> <p>También debe de guardar correspondencia el control de versiones de las tablas en las fases de educación, ilación y especificación; esto implica que si el aplicativo detecta una modificación en la especificación debe lanzar un mensaje al usuario indicando la existencia de posibles efectos colaterales en las tablas de ilación y educación.</p>

Tabla 3.46: Plantilla de especificación de requisitos 0015

Atributos	Descripción
Código	ESP-0016
Nombre	Agregar un requisito no funcional
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>Requisito no funcional es una sección del sistema, en ella se registrarán todos los datos que involucran a un requisito no funcional como son datos que el usuario llena o selecciona:</p> <ul style="list-style-type: none"> • Nombre del requisito a especificar. • Datos del actor especialista (nombre y cargo). (Selegionable de entre los registrados en la organización desarrolladora definida en la creación del proyecto). (Si el actor tuviera más de un cargo el sistema nos pedirá elegir uno, de lo contrario se mostrarán todos). • Descripción de la especificación. • Comentarios (Opcional). Datos que el sistema completa automáticamente. • Código del requisito a especificar (Respetando la forma definida además de incremental). • Versión de la educación (todo requerimiento inicia en 0.1 al ser creado). • Datos de la especificación FECHA (Fecha cuando se educe el requerimiento). <p>Precondición: el usuario deberá estar ejecutando el programa principal. En el sistema deben de estar creados los especialistas que intervienen en la ilación. El usuario debe de estar dentro de algún proyecto. El usuario debe de estar dentro del libro de RNF o desde el menú de herramientas (opcional). Postcondición: el sistema guardará el RNF en el proyecto.</p>
Prioridad	Alta
Dependencia	IL-0006
Comentarios	<p>Se implementará un código autoincremental para que en cada proyecto este detecte el código del último requisito dentro del libro correspondiente, por eso es automático, propio del sistema.</p> <p>El actor deberá aparecer en una lista desplegable si se encuentra registrado, la búsqueda por coincidencias ayudará en el caso de ser muchos los participantes. El usuario registrará en comentarios todos los datos que se relacionen con el requisito hasta el más mínimo detalle, pues puede servir posteriormente en la construcción y diseño.</p>

Tabla 3.47: Plantilla de especificación de requisitos 0016

Atributos	Descripción
Código	ESP-0017
Nombre	Modificar un requisito no funcional
Versión	V1.0
Autor	AUT-0001
Fuente	FUE-0001
Tipo	Funcional
Descripción	<p>El usuario podrá modificar un requisito con la sola acción de un botón que le permitirá editar los campos de la plantilla, al finalizar el usuario podrá guardar los cambios en cuyo caso se guardará sobre la versión en que se esté trabajando o versionar el requisito que permitirá al usuario ingresar una nueva versión y guardarla. Solo es necesario mostrar una lista de requisitos no funcionales sin precisar su clasificación.</p> <p>Precondición: el requisito a modificar debe de existir, por lo menos en su versión primaria.</p> <p>Postcondición: el sistema guardará el requisito en la tabla correspondiente.</p>
Prioridad	Alta
Dependencia	ILA-0006
Comentarios	<p>La modificación permitirá editar los campos de cada requisito, bloqueando los datos del código del requisito. Recordar que las tablas en las fases de educación, ilación y especificación forman parte de los requisitos funcionales y los requisitos no funcionales son aquellos requisitos que afectan al aplicativo desde un aspecto exterior como por ejemplo el desempeño algorítmico, el lenguaje de programación entre otros.</p> <p>La clasificación de los requisitos no funcionales no son parte del presente aplicativo, solo se necesita un listado de los requisitos no funcionales y su posible relación con las tablas de educación, ilación o especificación.</p> <p>Se pueden agregar tantas tablas de requisitos no funcionales como crea por conveniente el analista; la idea principal es agregarlas al aplicativo con la finalidad de contenerlas. El requisito no funcional no guarda relación con las etapas de educación, ilación y especificación.</p>

Tabla 3.48: Plantilla de especificación de requisitos 0017

Finalmente, y tomando en cuenta todas las tablas elaboradas (lo expuesto es una muestra de todos los cuadros que corresponden al catálogo de requisitos para este problema) en las tres etapas correspondientes, la trazabilidad realizada se define en la tabla 3.49.

Educación (Tabla)	Ilación (Tabla)	Especificación (Tabla)
1	1	1, 2
	2	3, 4
	3	5, 6, 7, 8, 9, 10, 11
	4	12, 13, 14, 15, 16, 17, 18
	5	19, 20, 21, 22, 23, 24, 25
	6	26, 27, 28, 29, 30, 31, 32
2	7	33, 34, 35, 36
	8	37, 38, 39, 40
	9	41, 42, 43, 44
	10	45, 46, 47, 48
3	11	49, 50, 51
	12	52
4	13	53

Tabla 3.49: Trazabilidad entre requisitos

La tabla 3.49 se puede interpretar que la tabla 1 de la etapa de educación tiene relación con la tabla 1 en la etapa de ilación y con las tablas 1 y 2 de las tablas de especificación. Si se detecta un error o se considera hacer un cambio en un requisito específico, la tabla nos proporciona las relaciones y solo visitamos las tablas que guardan relación para llevar a cabo la actividad correspondiente de modificación.

Es necesario mencionar que cuando se utilicen las plantillas propuestas, estas no deben contener campos sin completar. Si no se tuviere algo que redactar se debe colocar la palabra "Ninguno".

CAPÍTULO 4

Educción de Requisitos

4.1 Introducción

La educación de requisitos es la primera etapa para elaborar el catálogo de requisitos. Según la RAE significa “acción y efecto de educir” y educir implica “sacar algo de otra cosa, deducir”. Ello implica la acción de obtener información del cliente y entender lo que vierte como concepto y criterio propio del entendimiento del proceso de su modelo del negocio.

La relación con el cliente o usuario debe ser cordial con la finalidad de que el analista logre entender y comprender las necesidades de este, pero traduciéndolas en términos que entienda el propio analista manteniendo los criterios del cliente. Esta actividad, considerada una de las más complejas, debe ser llevada con la seriedad del caso y terminado por documentar hechos o sucesos que surgen durante el trayecto.

Es necesario emplear o adaptar técnicas que permitan una adecuada relación con el cliente con la finalidad de resolver el problema del entendimiento de estas necesidades y el comienzo de una escala de preguntas y visitas que concluyan con obtener la verdadera información sobre la situación de los procesos de una organización. Esta gestión de la información permite ordenar la forma de cómo se debe de percibir la elaboración de las plantillas de trabajo.

4.2 ¿Qué es la educación de requisitos?

Es la actividad que consiste en obtener información con respecto a las necesidades del cliente y entender, analizar y deducir los aspectos centrales para poder convertirlos en necesidades del analista y codificador (requisito).

4.3 Proceso de educación de requisitos

La figura 4.1 indica que la educación de requisitos comienza con una solicitud de información al cliente para luego concertar las correspondientes entrevistas. En este conjunto de entrevistas se descubren una serie de incidentes los mismos que son analizados para luego unirlos e integrarlos en conceptos unificados; finalmente ser interpretados hasta entender el modelo del negocio. Si se encuentran inconsistencias, se corrigen para nuevamente volver a realizar el análisis; este proceso se repite tantas veces hasta finalmente entender el modelo del negocio.

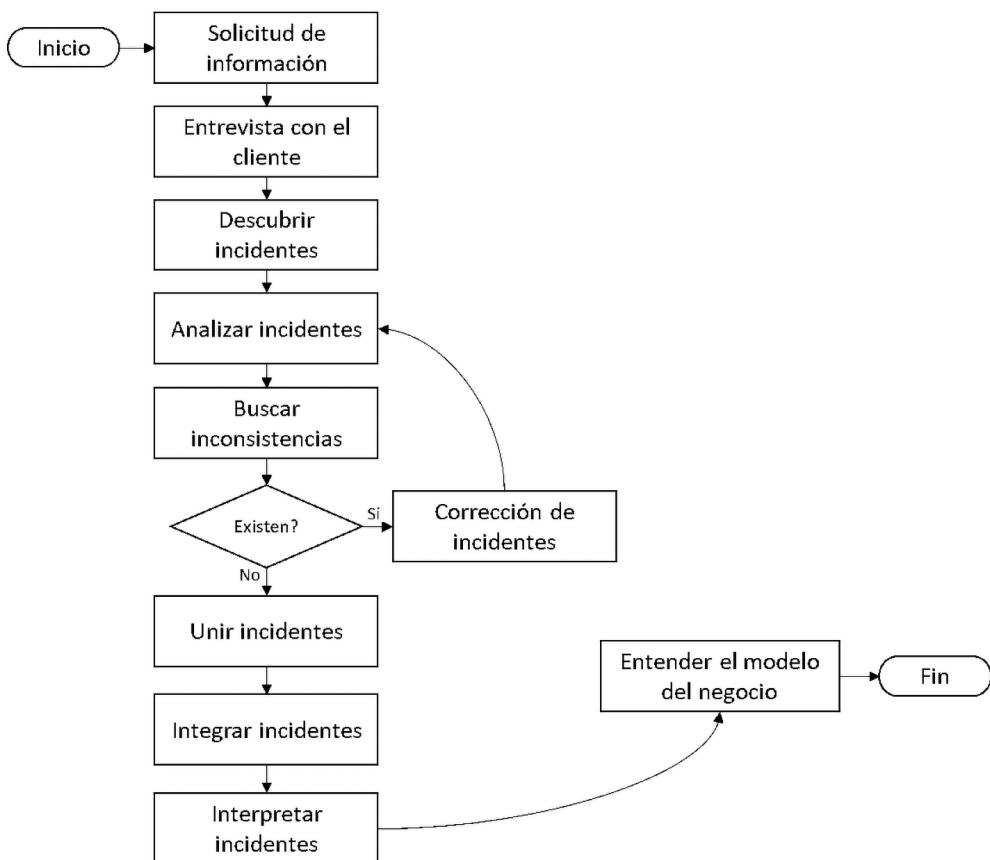


Figura 4.1: Proceso de educación de requisitos

Es probable que este proceso tarde bastante tiempo debido a que existen un conjunto de factores externos para que esto suceda como, por ejemplo: el idioma, las expresiones empleadas por el cliente, el desconocimiento del contexto por parte del analista, entre otros. El analista debe insistir en los incidentes, analizarlos hasta terminar de comprenderlos y para ello debe concretar tantas entrevistas como sea posible.

4.4 Técnicas para la educación de requisitos

Muchos investigadores han tratado este tema con diversos niveles de profundidad. A continuación, se presenta un resumen de las técnicas que se emplean para educacionar requisitos entre las que se encuentran las técnicas de alto y bajo nivel:

1. Diseño de aplicación conjunta (JAD). Técnica que permite la colaboración entre todos los actores del sistema permitiendo una visión compartida del mismo. Su principal actividad es la dinámica de grupo que permite obtener requisitos en función de la opinión de sus actores.
2. Entorno de bucles adaptativos. Consiste en establecer un entorno colaborativo entre usuarios y desarrolladores para educir requisitos por medio del aprendizaje. Los usuarios se comportan como los actores más importantes.
3. Prototipos. Técnica que permite al usuario entender los procesos en función de interfaces gráficas de usuario. No se debe de emplear la generación de interfaces de las plataformas de los lenguajes de programación porque induce al usuario a entender que lo que se le muestra es el producto final.
4. Entrevistas. Consiste en conversar, de manera sostenible, con los usuarios del sistema. Su finalidad es lograr educacionar los requisitos de manera clara y objetiva bajo un ciclo de retroalimentación adecuado.
5. Cuestionarios. Técnica que implica educacionar los requisitos por medio de un conjunto de interrogantes previamente diseñadas. Su desventaja se encuentra en la redacción de las respuestas ya que pueden introducir inconsistencias o ambigüedades.
6. Lluvia de ideas. Técnica que consiste en generar ideas por medio de sesiones de un grupo de trabajo, aproximadamente de 4 a 6 integrantes. Es útil cuando se trata de gestar un conjunto de puntos de vista con respecto a una idea.
7. Arreglos-Q. Es una técnica que consiste en elaborar un conjunto de tarjetas en función de postulados que permiten educacionar los requisitos. Para lograr el objetivo se definen arreglos que guardan estrecha relación con la distribución normal.
8. Análisis de mercado. Esta técnica se encuentra orientada a recoger información de productos adquiridos para predecir su comportamiento futuro. Identificando las características de un producto se puede predecir el comportamiento de otro similar.
9. PIECES. Facilita esquemas para el proceso de educación cuando los analistas no tienen la experiencia del caso. La técnica se encuentra desarrollada para resolver problemas de rendimiento, información y datos, economía, control, eficiencia y servicios. De aquí deriva el acrónimo de esta técnica.
10. Análisis de factores críticos. Consiste en determinar e identificar los factores críticos de un sistema; a partir de ellos se determinan los factores relevantes para terminar formulando el verdadero requisito. Es bastante empleado ante la existencia de abundante tecnicismo.

11. STROBE, Structure Observation of the Environment. Técnica que consiste en llevar a cabo un completo análisis de la organización. Identifica las áreas que se encuentran involucradas en la educación de requisitos y trabaja con ellas para definirlas de manera clara.
12. Análisis jerárquico de tareas. Consiste en llevar a cabo un análisis de lo que se requiere del usuario. El análisis se encuentra en función de procesos con la finalidad de entender el comportamiento del negocio.
13. Casos de uso. Esta técnica consiste en representar el modelo del negocio en plantillas de casos de uso. Estos casos de uso deben estar libre de inconsistencias para no introducir efectos colaterales en otros requisitos.
14. Escenarios. Técnica que consiste en entender el flujo de información a lo largo de un conjunto de áreas integrantes de un contexto general. Con esta consigna se pueden identificar los procesos transversales.
15. Guion gráfico. Ampliamente conocido como storyboards. Esta técnica consiste en presentar una secuencia gráfica de la forma como se desplegarán las interfaces gráficas de usuario con el único objetivo de hacer comprender al usuario como es la estructura del producto que se está construyendo.
16. Análisis competitivo. Técnica que consiste en entender a la empresa y la relación con su entorno; su finalidad es determinar los probables cambios, el posible impacto y adaptación ante ellos.
17. Observación directa. Técnica que consiste en observar directamente los procesos o procedimientos de la organización. Lleva a entender directamente el problema del flujo de información.
18. Investigación contextual. Técnica que acota el sector donde se lleva a cabo el trabajo; otorgando la posibilidad de obtener características particulares apropiadas del área acotada. Esta técnica influye en los objetivos de la construcción del producto.
19. Organización de conceptos. Técnica que consiste en reunir a las personas, dentro del área de la organización, que conocen criterios y flujos de la información. Son las personas que más conocen con respecto a los procesos internos.
20. Mapas de roles. Técnica que consiste en analizar los roles de cada uno de los integrantes de un área de la organización con la finalidad de entender el modelo del negocio. Esta técnica permite dinamizar la información de entrada con la finalidad de obtener buenos resultados.
21. Perfil de usuarios. Técnica que consiste en preparar el software en función del nivel de actitud y conocimiento que presenta el usuario. Los cambios sugeridos por el usuario no afectan al resto de ellos.
22. Etnografía. Técnica que lleva a cabo un análisis de las prácticas culturales de los usuarios. El estudio más directo es el de analizar las costumbres con respecto a las prácticas de uso de la información.

23. Puntos de vista. Técnica que emplea las opiniones de los usuarios sobre la situación de algún área con respecto a la información. Estas opiniones pueden ser controversiales, pero producto de su análisis puede ser de gran ayuda para el analista.

4.5 Desventajas de la educación de requisitos

El arte de educacionar requisitos puede ser obstruido por un sin número de problemas o inconvenientes de los cuales debemos de tener cuidado para lograr elaborar un catálogo adecuado. Entre sus desventajas podemos mencionar las siguientes:

- Si el contexto cultural del usuario es diferente al del analista entonces existe una tendencia a no entregar información completa.
- El tecnicismo que emana del analista puede lograr espantar al usuario permitiendo que este no aporte a las necesidades de información del analista.
- Muchas veces la forma de expresarse pueden ser un obstáculo para la comunicación. Las expresiones despectivas o egoístas terminan por alejar al usuario.
- La puntualidad es otro factor importante. Si el analista es impuntual se genera una mala percepción sobre la calidad del trabajo del mismo.
- La falta de concentración en el modelo del negocio hace que el usuario sea renuente en la nueva entrega de información.
- El excesivo uso de un modelo, técnica, metodología o método hacen perder el criterio del analista en entender el modelo del negocio.

4.6 Ejemplo de educación de requisitos

El ejemplo que se propone corresponde al “Sistema de Inventarios de la Bodega ELVITA”, sistema elaborado por un grupo de estudiantes de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa en el marco del curso de Ingeniería de Requerimientos. El objetivo principal del proyecto fue el de gestionar los inventarios de la bodega, así como el de enseñar a los estudiantes como se debe de elaborar un catálogo de requisitos.

A continuación, se muestran las tablas de la fase de educación no sin antes resaltar que las plantillas empleadas fueron diseñadas de acuerdo a las necesidades del usuario y de la complejidad del producto a construir. Esta muestra una ligera modificación con respecto a las plantillas proporcionadas en el texto.

Código educación	EDU-0001
Nombre	Gestión de Usuarios
Versión	01.02
Fecha	25/05/2020
Autor	AUT-0002
Actor	ACT-0001
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0001,ILA-0002,ILA-0003,ILA-0004
Descripción	El cliente manifestó que las personas que usarán el sistema son únicamente su esposa, su ayudante y él. Además se debe: <ul style="list-style-type: none"> • Registrar usuario. • Ingresar al sistema a través del nombre de usuario en turno. • Modificar datos de usuario. • Deshabilitar o Habilitar (cambiar de estado).
Importancia	Vital
Estado	Concluido
Comentario	El presente requisito tuvo origen en las preguntas 1 y 5 de la FUE-0004: 1. ¿Podría usted describir en qué consiste su negocio? 5. ¿Cuántas personas trabajan con usted? El presente requisito tuvo origen en la pregunta 2 de la FUE-0007: 2. ¿Le gustaría ingresar a más usuarios posteriormente o solo 2?

Tabla 4.1: Requisito de educación 0001

En una mirada al campo “versión” podemos notar que cuenta con el valor 01.02 que implica haber realizado varias iteraciones antes de concebir la tabla de educación final. Haciendo una revisión de las actas generadas por cada entrevista se advierte que fueron varias entrevistas antes de obtener la misma. Esto implica la inexistencia de claridad del modelo del negocio para poder definir la tabla EDU-0001.

Código educación	EDU-0002
Nombre	Gestión de Categorías
Versión	01.03
Fecha	29/05/2020
Autor	AUT-0008
Actor	ACT-0001 ACT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0005,ILA-0006,ILA-0007,ILA-0008,ILA-0009
Descripción	El cliente manifestó que en su bodega se venden productos de abarrotes, frutería y verduras, artículos de belleza y un poco de librería. Por lo tanto, será de suma conveniencia manejar categorías para separar los distintos productos que puede llegar a tener en la bodega. Además, que el cliente manifestó que desea gestionarlas. Mencionó: <ul style="list-style-type: none"> • Dentro de una categoría podemos encontrar distintos productos que tengan relación entre sí. • Generar una lista de categorías. • Crear nuevas categorías. • Modificar los datos de la categoría. • Eliminar una categoría (Siempre y cuando no se tengan productos asignados).

Importancia	Vital
Estado	Concluido
Comentario	El presente requisito tuvo origen en la pregunta 2 de la FUE-0004: 2. ¿Qué productos vende usted en su negocio? y en la pregunta 3 de la FUE-0007: 3. ¿Le gustaría administrar sus categorías o está bien tener solo 3?

Tabla 4.2: Requisito de educación 0002

Código educación	EDU-0003
Nombre	Gestión de Productos
Versión	01.03
Fecha	11/06/2020
Autor	AUT-0004
Actor	ACT-0001 ACT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0010,ILA-0011,ILA-0012,ILA-0013,ILA-0014,ILA-0015
Descripción	EL cliente manifestó vender los siguientes productos: en abarrotes, en frutas, frutería, verduras, artículos de belleza y un poco de librería, yogurt, leche, más que nada embutidos. De ello se desprenden los siguientes requisitos: <ul style="list-style-type: none"> • Gestionar productos (crear, modificar, eliminar, buscar). • Crear producto (datos: nombre, unidad de medida y descripción). • Modificar producto (nombre, unidad de medida y descripción). • Eliminar producto (siempre y cuando no haya cantidades registradas / lógica). • Buscar un producto de una lista de productos. • Ver un producto. • Establecer las cantidades y el precio de los productos en stock (una vez registrado poder inicializar el producto con una cantidad, esto se da al inicio del registro de un producto nuevo). • La precisión de datos numéricos no excederá a dos decimales.
Importancia	Vital
Estado	Concluido
Comentario	Origen del requisito: preguntas 2 de la FUE-0004: 2. ¿Qué productos vende usted en su negocio? y en las preguntas 11,14,25,27,43 y 44 de la FUE-0007: 11. ¿Productos con mayor cuidado por fecha de vencimiento? 14. ¿Cuál es la forma que se agrupen los productos para ver los stocks? 25. ¿Cuán a menudo puede tener nuevos productos en tienda? 27. ¿Cuántos tipos de productos tiene actualmente en su tienda? 43. ¿Cómo maneja los precios en su empresa? (0.50 o 0.55) 44. ¿Desea ver valores de productos en dos decimales o más?

Tabla 4.3: Requisito de educación 0003

Código educación	EDU-0004
Nombre	Gestión de Proveedores
Versión	01.03
Fecha	11/06/2020

Autor	AUT-0007
Actor	ACT-0001 ACT-0002
Fuente	FUE-0007
Código ilación	ILA-0016,ILA-0017,ILA-0018,ILA-0019,ILA-0020 eILA-0021
Descripción	<p>El cliente, manifestó la necesidad de lo siguiente:</p> <ul style="list-style-type: none"> • Administrar los proveedores (crear, modificar, eliminar y ver). • Organizar productos en base a los proveedores. • Asignar a proveedores a categorías y productos.
Importancia	Vital
Estado	Concluido
Comentario	<p>Origen del requisito son las preguntas 16, 24, 26, 28 y 29 de la FUE-0007:</p> <p>16. ¿Le es relevante organizar por proveedores?</p> <p>24. ¿Suele trabajar con los mismos proveedores?</p> <p>26. ¿Le gustaría poder administrar proveedores?</p> <p>28. ¿Cuán a menudo tiene nuevos proveedores?</p> <p>29. ¿En qué categoría tiene proveedores no definidos?</p>

Tabla 4.4: Requisito de educación 0004

Código educación	EDU-0005
Nombre	Gestión de Alarmas
Versión	01.02
Fecha	11/06/2020
Autor	AUT-0001
Actor	ACT-0001 ACT-0002
Fuente	FUE-0007
Código ilación	ILA-0022,ILA-0023,ILA-0024,ILA-0025,ILA-0026
Descripción	<p>El cliente manifestó le gustaría lo siguiente:</p> <ul style="list-style-type: none"> • Crear, modificar y eliminar una alarma dada una condición. • Le gustaría notificaciones acerca del stock de productos.
Importancia	Vital
Estado	Concluido
Comentario	<p>El presente requisito tuvo origen en las preguntas 4 y 5 de la FUE-0007:</p> <p>4. ¿Qué tipo de notificaciones les gustaría?</p> <p>5. ¿Le gustaría una alerta para saber qué productos deben rotar?</p>

Tabla 4.5: Requisito de educación 0005

Código educación	EDU-0006
Nombre	Gestión de Abastecimiento
Versión	01.03
Fecha	11/06/2020
Autor	AUT-0003
Actor	ACT-0001 ACT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0027,ILA-0028,ILA-0029,ILA-0030,ILA-0031,ILA-0032

Descripción	El cliente manifestó tener las siguientes actividades diarias: "Me levanto a tempranas horas de la mañana para ir a hacer las compras, para abastecer los productos que hacen falta, de diario (...) "se hacen compras de productos que hacen falta en la tienda, se recibe los productos que llegan de los proveedores (...)" . De ello se desprenden los siguientes requisitos: <ul style="list-style-type: none">• Generar una lista de "productos a comprar del dia/semana".• Registrar compras.• Modificar los datos de la compra.• Afectar las cantidades y el precio de los productos en stock.• Registrar los nuevos productos.• Eliminar una compra.
Importancia	Vital
Estado	Concluido
Comentario	La entrada de datos numéricos se limita a dos decimales. Ej. 12.50 El presente requisito tuvo origen en las preguntas 2,4 y 9 de la FUE-0004: 2. ¿Cuenta Ud. con algún programa informático para manejar su negocio o alguna parte de su negocio? 4. ¿Qué actividades considera Ud. que son críticas en su negocio? 9. ¿Qué días puede atendernos para próximas entrevistas? y en las preguntas 6 y 7 de la fuente FUE-0007? 6. ¿Cómo se abastece?, generalmente como es que ingresan sus productos 7. ¿Realiza sus abastecimientos en una sola compra?

Tabla 4.6: Requisito de educación 0006

Código educación	EDU-0007
Nombre	Gestión de Salidas
Versión	01.02
Fecha	28/05/2020
Autor	AUT-0005
Actor	ACT-0001 ACT-0002
Fuente	FUE-0007
Código ilación	ILA-0033,ILA-0034,ILA-0035,ILA-0036
Descripción	El cliente manifestó la necesidad de lo siguiente, la salida del inventario no es únicamente por las ventas. Con esto se define: <ul style="list-style-type: none">• Registro de salida extraordinaria (consumo propio, robo, productos vencidos).• Registro de salida ordinaria (venta).• Ver registro de salidas.• Modificación de salidas.• Eliminar salidas.
Importancia	Vital
Estado	Concluido
Comentario	La entrada de datos numéricos se limita a dos decimales. Ej. 12.50. El presente requisito tuvo origen en las preguntas 8, 9, 10 y 36 de la FUE-007: 8. ¿Qué tratamiento les da a los productos que consume su familia o personas relacionadas con ellas? 9. ¿Cada cuánto tiempo suceden los robos o sustracciones de objetos o productos? 10. ¿Cada cuánto se deshace de productos vencidos? ¿Una vez vencidos cuánto tiempo los conservaba? 36. ¿Cuántas salidas y/o entradas tiene usted en forma diaria o semanal o mensual?

Tabla 4.7: Requisito de educación 0007

Código educación	EDU-0008
Nombre	Gestión de Reportes
Versión	01.04
Fecha	11/06/2020
Autor	AUT-0006
Actor	ACT-0001 ACT-0002
Fuente	FUE-0004, FUE-0007, FUE-0008
Código ilación	ILA-0037,ILA-0038,ILA-0039,ILA-0040
Descripción	<p>El cliente manifestó tener las siguientes actividades diarias: "Me levanto a tempranas horas de la mañana para ir a hacer las compras, para abastecer los productos que hacen falta, de diario (...) trabajo con diferentes proveedores (...) si ya van a vencer, anticipadamente una semana antes se puede cambiar el producto (...) si todo está en orden (...) me gustaría más información (...) aceite, por un lado, menestras en otro lado (...) más por proveedores (...) por una categoría en general (...) todo redondeado y los precios siempre son exactos (...) en dos decimales (...) que quede guardado y se pueda borrar (...)" Se dan los siguientes requisitos:</p> <ul style="list-style-type: none"> • Generar un reporte de abastecimiento de productos a comprar en base a la cantidad de stock (p. ej. "productos que requieren comprarse en el del día/semana"). • Generar un reporte de proveedores de los productos. • Generar un reporte de usuarios del sistema. • Generar un reporte de las alertas del sistema. • Generar un reporte de las categorías existentes en el sistema. • Generar reportes de productos con personalización de filtros, considerando: fecha de caducidad, categoría, fecha de entrada, fecha de salida, ordenados por la cantidad de salidas, ordenados por la cantidad de stock disponible, proveedor, usuario. • Los precios deben ser redondeados a dos decimales. • Los reportes generados se guardan y se puedan borrar.
Importancia	Vital
Estado	Concluido
Comentario	<p>El requisito se originó a raíz de las preguntas 4 y 7 de la FUE-0004:</p> <p>4. ¿Qué actividades realiza en su negocio? 7. ¿Usted trabaja con varios proveedores o realiza las compras? y en las preguntas 10,11,12,13,14,15,16,17, 43, 44 de la FUE-0007: 10. ¿Cada cuánto se deshace de productos vencidos? 11. ¿Qué productos debe tener cuidado debido a la fecha de vencimiento? 12. ¿Qué información le gustaría tener al iniciar sus operaciones del día? 13. ¿Es suficiente con ver una lista de todos los stocks por productos? 14. ¿Cuál es la forma que se agrupen los productos para ver los stocks? 15. ¿Qué tipos de datos les gustaría tener en el reporte? 16. ¿Le es relevante organizar por proveedores? 17. ¿Quisiera tener un reporte de detalle por producto o por categoría? 43. ¿Cómo maneja los precios en su empresa?, (0.50 o 0.55) 44. ¿Le gustaría ver el costeo de sus productos en dos decimales o más? y en la pregunta 3 de la FUE-0008: 3. ¿Los reportes se almacenarán?</p>

Tabla 4.8: Requisito de educación 0008

Código educación	EDU-0009
Nombre	Gestión de Estadísticas
Versión	01.05
Fecha	29/05/2020
Autor	AUT-0003
Actor	ACT-0001 ACT-0002

Fuente	FUE-0007, FUE-0008
Código ilación	ILA-0041,ILA-0042,ILA-0043,ILA-0044,ILA-0045
Descripción	De acuerdo al análisis de las respuestas del cliente: "si son productos redonditos, costos válidos", se desprende lo siguiente: <ul style="list-style-type: none"> • Estadística de Productos dada una categoría por ventas. • Torta producto por categoría, vencidos / por vencer (3 semanas) / vigentes. • Gráfico estadístico "Salidas" (de acuerdo al motivo de salida). • Ventas por categoría (tramo de tiempo).
Importancia	Descargar estadística (Gráfico y su tabla generadora).
Estado	Concluido
Comentario	El presente requisito tuvo origen en las preguntas 18, 19, 20, 21, 22, 23 de la FUE-0007: 18. ¿Qué tipo de gráficos le gustaría ver con respecto a los movimientos de sus inventarios? 19. En cuanto a los productos ¿le sería relevante ver en un gráfico que productos se han vendido más? 20. ¿Le sería relevante poder ver en un gráfico de qué productos no tienen movimiento en la tienda? 21. ¿Le sería relevante poder ver como es el histórico del costo promedio de cada producto? 22. ¿Le sería relevante tener un estadístico que le permite saber cuánto stock mínimo por producto debería tener? 23. ¿Le gustaría saber cuánto es el promedio de stock por producto en cuanto al histórico de sus ventas? y de las preguntas 1 y 2 de la FUE-0008: 1. ¿Qué tipo de gráficos le gustaría ver como estadísticas? 2. ¿Los costos tendrían que ser en un gráfico de tiempo?, ¿Cuál le gustaría?

Tabla 4.9: Requisito de educación 0009

Se debe notar la forma limpia cómo se construye la educación y las interrogantes al cliente. El resultado es la asociación de las preguntas con los requisitos permitiendo una construcción que comienza a "*limpiar el camino del entendimiento del modelo del negocio*". Esta forma de educacionar requisitos le proporciona una estrategia al analista para entrelazar los mismos y encontrar errores en etapas tempranas de la construcción.

Otro detalle importante es la relación propiamente dicha entre tablas de las diferentes etapas. Notar que la fila "*Código de ilación*" implica una asociación con las tablas de la etapa de ilación. De esta manera se forma el enlace permitiendo lograr una adecuada trazabilidad y facilitar el hallazgo de errores de manera encadenada.

La idea de mantener al "*autor*", "*actor*" y "*fuente*" implica conocer quién confeccionó la tabla, qué elemento del sistema interviene y quién proporcionó la correspondiente información. Ante un error, se sabe a quién recurrir para su solución final sin olvidar de versionar la correspondiente tabla. Se debe tener en cuenta que los códigos de ilación se agregan una vez que se tengan confeccionadas las tablas de la etapa de ilación completando las tablas de la etapa de educación en una nueva versión.

CAPÍTULO 5

Ilación de Requisitos

5.1 Introducción

La Ilación de Requisitos es la segunda etapa en la elaboración del catálogo de requisitos e implica como el analista interpreta el modelo del negocio. Según la RAE, significa “acción y efecto de inferir una cosa de otra”; esto significa entender el modelo del negocio del sistema del cliente y transformarlo, bajo criterios personales, en elementos que posteriormente puedan ser traducidos en la codificación.

Una vez que el analista ha deducido los módulos que conforman el producto a construir pasa a comprender esta lógica que le proporciona el modelo del negocio. Esta etapa es iterativa ya que si no logra comprender algún componente de esta etapa debe recurrir a la etapa de educación para determinar lo que se necesita, lo que también implica reuniones con el cliente hasta entender el modelo del negocio entregado por el cliente.

Una tabla en la etapa de educación puede contener una o más tablas en la etapa de ilación. Esta relación permitirá a futuro mantener una adecuada trazabilidad, la misma que permite una buena retroalimentación ahorrando tiempo y costo en el mantenimiento de estas tablas. Las técnicas a emplear en esta fase dependen de la complejidad del problema a resolver y de la experiencia del analista que realiza esta tarea.

5.2 ¿Qué es la ilación de requisitos?

Es una actividad que consiste en transformar los conocimientos obtenidos del entendimiento del modelo del negocio proporcionado por el cliente en esquemas propios de entendimiento del analista permitiéndole esquematizar y reforzar, en un futuro, la arquitectura del software para lograr una adecuada construcción del producto.

5.3 Proceso de ilación de requisitos

Esta etapa se encuentra relacionada con el entendimiento, de parte del analista, de la realidad a la que se enfrenta. La figura 5.1 indica que el proceso de ilación tiene como entrada al documento de educación. Las tablas de educación son analizadas por el analista con la finalidad de comprenderlas y permitirle descubrir un conjunto de incidentes. Estos son analizados para saber si existe una correspondencia con la realidad; si esta existe entonces se lleva a cabo la ilación de requisitos caso contrario se debe de interpretar estos incidentes.

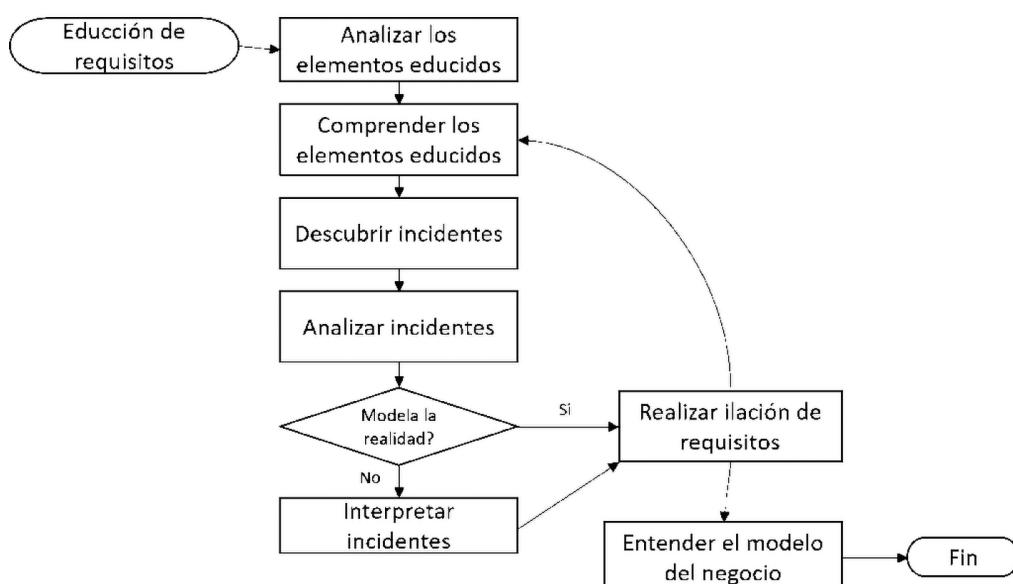


Figura 5.1: Proceso de ilación de requisitos

Luego de un proceso iterativo, se llega a entender el modelo del negocio y para lo cual se va definiendo el documento de ilación de requisitos. El proceso culmina con un real conocimiento del modelo del negocio a automatizar, de las fortalezas y debilidades del sistema y del proceso para la especificación de requisitos de software; permitiendo un entendimiento sobre el comportamiento en la etapa de codificación del producto.

5.4 Técnicas para la ilación de requisitos

Estas técnicas presentan una mayor especificación ya que no consiste en entrevistar al cliente, sino que a partir del documento de educación se entienda el modelo del negocio a automatizar incluido su dominio de acción. Es común emplear las siguientes técnicas:

1. Investigación presencial: Técnica que consiste en que el analista lleve a cabo una comprobación, sobre las áreas más importantes, de la etapa de educación o en su defecto

un refrescamiento de lo entendido en la etapa previa. Este refrescamiento le permite entender la problemática y definir los elementos de ilación.

2. Observación de ambientes. Técnica que consiste en llevar a cabo, en forma presencial, la comprobación de la etapa de educación y la culminación de la etapa de ilación con criterios personales.

5.5 Desventajas de la ilación de requisitos

Las fallas que se pueden encontrar cuando se lleva a cabo esta tarea son las siguientes:

- La etapa de educación no se encuentra correctamente especificada de acuerdo al modelo del negocio.
- Los requisitos en la etapa de educación presentan inconsistencias y estas no han sido corregidas.
- Los requisitos expuestos en la etapa de educación no se encuentran validadas por el usuario.

5.6 Ejemplo de ilación de requisitos

El ejemplo expuesto es la continuación del catálogo de requisitos elaborado para el proyecto del Sistema de Inventarios. En esta parte se muestran las tablas de ilación.

La plantilla de ilación propuesta puede ser definida dependiendo del grado de granularidad con que se desee trabajar o de la complejidad del problema a resolver. Si se desea contar con información minuciosa entonces a la plantilla se pueden agregar tantos campos como ítems se deseen expresar. Por ejemplo, se pueden agregar las precondiciones, postcondiciones y otros campos con características que son necesarias para comprender el problema a resolver.

Un artefacto adecuado para plasmar toda esta información son los casos de uso o las tarjetas “Class Responsibility Collaborator”. Los casos de uso son más empleados por su facilidad de presentar y manipular la información y porque se pueden agregar tantas filas como se consideren necesarias para expresar las necesidades del cliente. La flexibilidad de este artefacto permite que el mismo se adapte a las necesidades del analista.

Otro detalle interesante es la forma como se encadenan las tablas de educación, de ilación y de especificación, incluido otros artefactos diseñados. En la plantilla de ilación se define un campo denominado “Código educación”, en el cual se detalla la relación con la tabla de educación. Esta forma de encadenamiento facilita la trazabilidad. Cuando se encuentra un error en una tabla de ilación entonces se observa de cuál tabla de educación procede; de esta manera se logra una corrección en cascada y en donde el control de versiones cobra relevancia.

En el campo "Comentarios" podemos agregar descripciones indispensables para terminar de entender el modelo del negocio o cualquier apunte necesario que implique relevancia hacia el entendimiento de los procesos o modelo del negocio. En este campo también se puede hacer referencia del interfaz gráfico del usuario asociado con la descripción de la tabla de ilación; inclusive se puede hacer alusión a diagramas de secuencias, diagramas de colaboración entre otros.

Elementos importantes como el control de versiones resuelve el problema de la administración de requisitos y el mantenimiento del historial de cambios de los mismos. La prioridad implica el grado de importancia que presenta el requisito para su automatización y bajo este criterio se puede determinar los requisitos que primeramente deben ser codificados e incluidos, así como de las dependencias entre ellos.

El flujo normal implica una descripción del procedimiento que sirve como punto de apoyo para la definición de algoritmos y la influencia de los requisitos no funcionales. En este punto, si fuere necesario, se pueden agregar las precondiciones o las postcondiciones. Si no fuera necesario, basta con un simple relato introducido en los comentarios para resolver la problemática.

Otro tema importante son las "Precondiciones" y las "Postcondiciones", en estos campos se agregan el conjunto de condiciones que no son atacadas por el procedimiento común pero que son importantes al momento de codificar el procedimiento ya que proporcionan las entradas y salidas del algoritmo.

Notar que en la línea de comentarios, se pueden entregar explicaciones que permitan el esclarecimiento del problema sobre el modelo del negocio investigado. Suele suceder que los analistas las denominen precondiciones o postcondiciones y estas pueden ser incluidas como líneas independientes en la plantilla de trabajo de tal manera que precisan una mejor formalización o apreciación del mismo.

Es importante la codificación de las tablas de educación porque de esta manera podemos localizar los errores con suma facilidad. El control de versiones también es otro tema importante ya que implica la cantidad de veces que la tabla ha sido refinada o ha tenido retroalimentación con la finalidad de aclarar los criterios de diseño. A partir de este control de versiones se pueden obtener métricas de calidad.

Para este problema es importante incluir la prioridad de la ilación porque de esta manera se guarda el orden de construcción desde el ángulo de vista de la codificación y su relación con los otros artefactos de diseño como lo son diagramas de clases, diagramas de objetos, diagramas de colaboración, diagramas de secuencias, entre otros; o aquellos diagramas dependiendo del lenguaje de modelado escogido. En el flujo normal de datos se define el procedimiento adecuado que conlleva a la elaboración del respectivo algoritmo.

Las tablas de ilación son la percepción que tiene el analista con respecto al problema y la debe

de refinar tantas veces como sea posible hasta encontrar los mejores criterios que lo lleven a entregar una adecuada solución. Esta solución va a asociarse con los primeros criterios de la especificación de los requisitos de software. Esto implica que el sistema a automatizar va tomando una estructura que nos acerca a una primera forma de codificación del producto y en donde pueden ser incluidos las interfaces gráficas del usuario o los componentes, así como los diagramas previstos por algún lenguaje de modelado.

Por ejemplo, la tabla 5.1 muestra la relación entre la tabla de ilación ILA-0001, la tabla de educación EDU-0001 y las tablas de especificación ESP-0001, ESP-0002, ESP-0003, ESP-0004 y ESP-0005. Las tablas muestran una clara trazabilidad que permite encontrar errores que se pudieran detectar en cualquiera de ellas. Esta tabla ha sido elaborada por el autor AUT-0002 y participan los actores ACT-0001 y ACT-0002. La información ha sido extraída de las fuentes FUE-0004 y FUE-0007. Al tener una versión con valor 1.05 implica que la información que representa ha sido complicada de obtener o en su defecto el analista no comprende el modelo del negocio en su totalidad.

Una característica que debemos notar es que muchas veces se escriben oraciones comunes entre tablas, por ejemplo: *"El siguiente procedimiento deberá tomarse en cuenta para obtener el resultado correcto..."*. Esta parte en común es importante porque a partir de ella se pueden determinar o encontrar patrones que tienen como objetivo el ahorro de tiempo. Estos patrones permitirán definir el diseño de interfaces gráficas de usuario [38].

Por otro lado, los patrones ontológicos de dominio ayudarán a definir y elaborar herramientas orientadas al diseño de interfaces graficas de usuario bajo la misma filosofía de la programación visual. Esta ayuda es sustancial porque permite ahorro en tiempo y costo de construcción del producto de software permitiendo una definición estable a lo largo del catálogo de requisitos.

Esto también genera una estabilidad en la documentación de los requisitos de software y produce una lectura legible y entendible a lo largo de la descripción de los requisitos de software; permite una transmisión de resultados bastante claros a lo largo del documento proporcionando esquemas estructurales y conceptuales bastante definidos.

Código Ilación	ILA-0001
Nombre	Registrar datos de usuario
Versión	01.05
Fecha	11/06/2020
Autor	AUT-0002
Actor	ACT-0001, ACT-0002
Fuente	FUE-0004, FUE-0007
Código educación	EDU-0001
Código especificación	ESP-0001, ESP-0002, ESP-0003, ESP-0004, ESP-0005

Descripción	El sistema permite al administrador registrar usuarios.
Precondición	<ul style="list-style-type: none"> En la base de datos del sistema debe haber una tabla "Usuario". El usuario debe haber iniciado sesión correctamente.
Procedimiento	<ul style="list-style-type: none"> Seleccionar del menú principal la opción "Gestión de usuarios". Presionar el botón "Registrar" del menú de opciones. El sistema despliega un formulario con los siguientes campos: "Nombres", "Apellidos", "DNI", "Contraseña", "Tipo de usuario", "Nro de Celular", "Correo electrónico", "Dirección", "Estado". El usuario ingresa por teclado los datos: "Nombres", "Apellidos", "Contraseña", "DNI", "Nro de Celular", "Correo electrónico", "Dirección". El usuario selecciona de una lista de opciones el "Tipo de usuario". El usuario selecciona un radio button denominado "Estado". Presionar el botón "Guardar". El sistema almacena el nuevo registro en la tabla "Usuario" de la Base de datos.
Postcondición	El sistema muestra una ventana de confirmación del registro.
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0009: "Formularios de Registro de datos de usuario" (la sigla CON hace referencia a las tres primeras letras de la palabra "CONVENCIÓN").
Importancia	Vital
Estado	Concluido
Comentario	<p>El usuario ingresa por teclado los siguientes datos:</p> <ul style="list-style-type: none"> Nombres, Apellidos, Dirección => [az-AZ] Celular => [0-9][9] DNI => [0-9][8] Correo electrónico: [a-zA-Z0-9.!#\$%&*+=?^_`{ }~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+) Contraseña => (?=\w*\d)(?=.\w*[A-Z])(?=.\w*[a-z])\S{8,16} <p>El usuario selecciona de una Lista de opciones:</p> <ul style="list-style-type: none"> Tipo de usuario => [1,2] (1= "Administrador", 2= "Vendedor") El usuario selecciona el radio button: Estado => [I, A] (I = "Inactivo", A = "Activo") <p>El formulario de registro cuenta con la validación del tipo de dato en cada uno de sus campos.</p> <p>El sistema realizará una consulta a la base de datos tabla "Usuario" comparando el DNI ingresado con los registros existentes para evitar duplicados. En caso de existir dicho usuario en la base de datos, el sistema devuelve el mensaje "El usuario ya existe".</p>

Tabla 5.1: Requisito de ilación 0001

Código Ilación	ILA-0002
Nombre	Ingresar al sistema a través del nombre de usuario en turno
Versión	01.06
Fecha	11/06/2020
Autor	AUT-0002
Actor	ACT-0001, ACT-0002
Fuente	FUE-0004, FUE-0007
Código educación	EDU-0001
Código especificación	ESP-0006, ESP-0007
Descripción	El sistema permite el inicio de sesión a los usuarios previamente registrados.
Precondición	<ul style="list-style-type: none"> El usuario debe estar registrado en la base de datos del sistema, tabla "Usuario". El estado del usuario en la base de datos debe ser "Activo".

Procedimiento	<ul style="list-style-type: none"> Ingresar al sistema. El sistema despliega un formulario de inicio de sesión con los siguientes campos: "Correo electrónico", "Contraseña". El usuario ingresa por teclado los datos: "Correo electrónico", "Contraseña". Presionar el botón "Iniciar". El sistema consulta la tabla "Usuario" de la Base de datos utilizando los datos ingresados en el formulario de inicio de sesión.
Postcondición	El sistema muestra una ventana de confirmación de inicio de sesión
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0001: "Pantalla de Inicio de Sesión"
Importancia	Vital
Estado	Concluido
Comentario	<p>El usuario ingresa por teclado los siguientes datos:</p> <ul style="list-style-type: none"> Correo electrónico: [a-zA-Z0-9.!#\$%&*+=?^_`{}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)* Contraseña => (?=\\w*[a-zA-Z]\\w*[a-zA-Z]\\w*[a-zA-Z]\\w*){8,16} <p>El formulario de inicio de sesión cuenta con la validación del tipo de dato en cada uno de sus campos.</p>

Tabla 5.2: Requisito de ilación 0002

Código Ilación	ILA-0003
Nombre	Modificar datos de usuario
Versión	01.07
Fecha	11/06/2020
Autor	AUT-0002
Actor	ACT-0001, ACT-0002
Fuente	FUE-0004, FUE-0007
Código educación	EDU-0001
Código especificación	ESP-0008, ESP-0009, ESP-0010
Descripción	El sistema permite al administrador modificar los datos de los usuarios registrados.
Precondición	<ul style="list-style-type: none"> El usuario debe estar registrado en la base de datos del sistema, tabla "Usuario". El Administrador debe haber iniciado sesión correctamente.
Procedimiento	<ul style="list-style-type: none"> Seleccionar del menú principal la opción "Gestión de usuarios". Presionar el botón "Modificar" del menú de opciones. El sistema hace una consulta a la tabla "Usuario" de la base de datos. El sistema despliega un formulario con los campos: "Nombres", "Apellidos", "DNI", "Contraseña", "Tipo de usuario", "Nro de Celular", "Correo electrónico" y "Dirección" los cuales contendrán los datos de usuario listos para su modificación. El usuario modifica por teclado los datos: "Nombres", "Apellidos", "DNI", "Contraseña", "Nro de Celular", "Correo electrónico", "Dirección" del formulario de registro. El usuario selecciona de una lista de opciones el "Tipo de usuario". Presionar el botón "Guardar". El sistema almacena las modificaciones en la tabla "Usuario" de la Base de datos.
Postcondición	El sistema muestra una ventana que confirma la modificación de datos.
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0010: "Formularios en general"
Importancia	Vital
Estado	Concluido

Comentario	<p>El usuario ingresa por teclado los siguientes datos:</p> <ul style="list-style-type: none"> Nombres, Apellidos, Dirección => [az-AZ] Celular => [0-9][9] DNI => [0-9][8] Correo electrónico: [a-zA-Z0-9.!#\$%&*+=^_`{}~-]+@[a-zA-Z0-9-]{2,}\.[a-zA-Z0-9-]{1,} Contraseña => (?=\\w*\\d)(?=\\w*[A-Z])(?=\\w*[a-z])\\S{8,16} <p>El usuario selecciona de un Listbox:</p> <ul style="list-style-type: none"> Tipo de usuario => [1,2] (1= "Administrador", 2= "Vendedor"). <p>El formulario de inicio de sesión cuenta con la validación del tipo de dato en cada uno de sus campos.</p>
------------	---

Tabla 5.3: Requisito de ilación 0003

Código Ilación	ILA-0004
Nombre	Deshabilitar o Habilitar usuario (cambiar de estado)
Versión	01.06
Fecha	11/06/2020
Autor	AUT-0002
Actor	ACT-0001, ACT-0002
Fuente	FUE-0004, FUE-0007
Código educación	EDU-0001
Código especificación	ESP-0011, ESP-0012
Descripción	El sistema permite al administrador habilitar o deshabilitar las cuentas de usuarios previamente registrados.
Precondición	<ul style="list-style-type: none"> El usuario debe estar registrado en la base de datos del sistema, tabla "Usuario". El Administrador debe haber iniciado sesión correctamente.
Procedimiento	<ul style="list-style-type: none"> Seleccionar del menú principal la opción "Gestión de usuarios". El sistema muestra el listado de usuarios que están registrados en la tabla "Usuario" de la base de datos del sistema. El Administrador selecciona un radio button denominado "Estado" para cada usuario. Presionar el botón "Guardar". El sistema almacena el nuevo estado del usuario en la tabla "Usuario" de la base de datos del sistema.
Postcondición	El sistema muestra una ventana que confirma el cambio del estado del usuario.
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario".
Importancia	Vital
Estado	Concluido
Comentario	<p>El usuario selecciona un radio button:</p> <ul style="list-style-type: none"> Estado => [I,A](I = "Inactivo", A = "Activo"). <p>La función deshabilitar no está disponible entre administradores del sistema, solo se aplica de administrador a usuario.</p>

Tabla 5.4: Requisito de ilación 0004

Bajo este criterio se logran establecer 45 tablas de ilación que guardan relación estricta con las tablas de educación expuestas en el capítulo anterior.

CAPÍTULO 6

Especificación de Requisitos

6.1 Introducción

La palabra “especificación”, de acuerdo con el diccionario de la RAE, significa: “*Información proporcionada por el fabricante de un producto, la cual describe sus componentes, características y funcionamiento*”. Se puede inferir que existe un mayor nivel de detalle en el conocimiento de una actividad, producto o información, el mismo que depende de la forma como gestiona su conocimiento el analista.

La tarea de especificar los requisitos comienza con la etapa de educación, continua con el refinamiento sucesivo del conocimiento del modelo del negocio producto de las sucesivas entrevistas con el cliente y los stakeholders en la etapa de ilación y se culmina con una especificación de la información orientada al codificador. En esta especificación se definen los procedimientos y los datos del aplicativo a construir.

En la especificación de requisitos se emplean las plantillas de casos de uso bajo una definición clara orientada a resolver la complejidad del problema y proporcionando los primeros atisbos de solución para los diagramas de los lenguajes de modelado. Aunque algunos datos se pueden obtener de las plantillas de ilación, las plantillas de especificación terminan de redondear la idea generando una retroalimentación a la fase de ilación.

6.2 ¿Qué es la especificación de requisitos?

La especificación de requisitos es una actividad que se encuentra encaminada a lograr la especificidad de la información orientándolo a la etapa de codificación. En ella se muestra información concreta y específica que apoya a la codificación como lo son los tipos de datos, rutinas y subrutinas, entre otros. Esta actividad se encuentra apoyada por los diagramas obtenidos en la fase de ilación.

6.3 Proceso de especificación de requisitos

Esta etapa se encuentra relacionada con una mayor especificación de información, orientada y definida con conceptos relacionados con la codificación. En la figura 6.1 se aprecia que el artefacto de entrada es el documento de ilación. Por cada requisito de ilación se lleva a cabo un análisis con la finalidad de saber si se ha logrado entender este. Comprendido su significado se transforma en requisitos de especificación donde se agregan definiciones como el tipo de datos, atributos y métodos, entre otros.

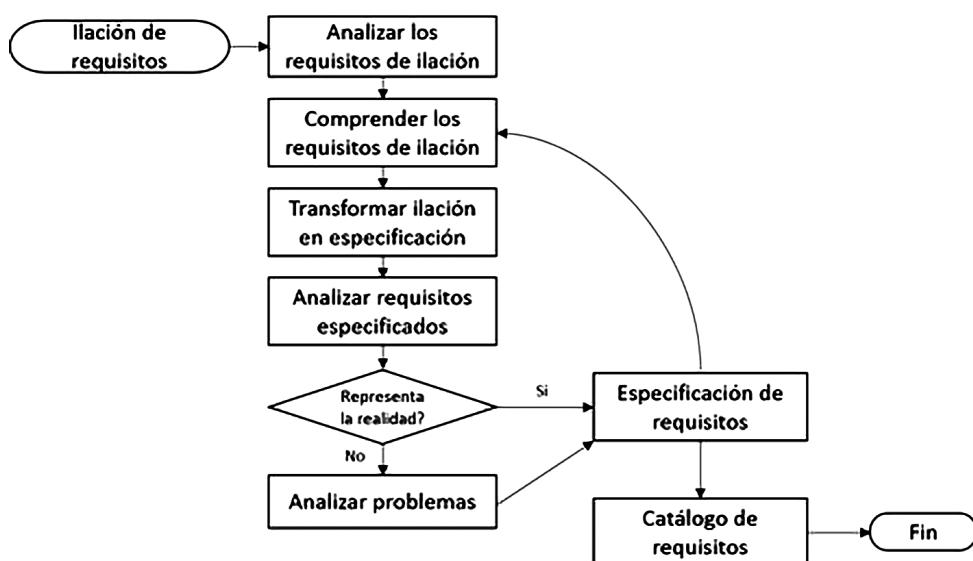


Figura 6.1: Proceso de especificación de requisitos

Después de la transformación se hace el análisis del requisito especificado; si cumple con la realidad entonces se almacena en el artefacto correspondiente, caso contrario se analiza el problema si no se ha logrado el cumplimiento. Después de sucesivas iteraciones se obtiene el catálogo de requisitos. Este artefacto se encuentra listo para continuar con el modelamiento del sistema en la etapa de diseño.

6.4 Técnicas para la especificación de requisitos

A continuación, se muestran las técnicas asociadas a la etapa de especificación de requisitos de software.

1. Descomposición funcional. Consiste en comprender los requisitos obtenidos en la etapa de ilación y recomponerlos en la etapa de especificación orientando el criterio hacia la codificación del producto. También en esta etapa se puede refinar el requisito generando una retroalimentación en las etapas anteriores.

2. Especificación de sentencias textuales. Consiste en describir textualmente los requisitos expresándolos en lenguaje natural. Esta técnica puede introducir más defectos en los requisitos porque se debe tener un control de la parte léxica, sintáctica y semántica.
3. Modelado del proceso. Permite que los analistas puedan diagramar los procesos del negocio a partir de los requisitos en cualquiera de sus tres etapas propuestas. La condicionante es que estos deben estar libres de inconsistencias o ambigüedades.
4. Modelado del dominio. Consiste en llevar a cabo un análisis con respecto al dominio de un sistema para lograr determinar los requisitos necesarios para la funcionalidad de las partes definidas.
5. Casos de uso. Ampliamente empleados para llevar a cabo el análisis de los requisitos en lenguaje natural. La plantilla puede ser definida dependiendo de la granularidad de la información o de la complejidad de la misma.
6. Checklist. Consiste en conceptualizar un conjunto de preguntas relacionadas con las funcionalidades del sistema permitiendo obtener, de esta manera, la especificación de los requisitos de software.
7. Inspección. Consiste en llevar a cabo una revisión de las interfaces gráficas de usuario y compararlos con lo expresado por la necesidad del cliente para determinar si el requisito cumple con las expectativas del caso.
8. Prototipos. Consiste en diseñar representaciones gráficas, conocidas comúnmente como interfaces gráficas de usuario, donde se representan los requisitos de software definidos por medio de las necesidades del cliente o del usuario final.

6.5 Desventajas de la especificación de requisitos

Las desventajas o falencias de la etapa de especificación de requisitos se muestran a continuación:

1. Se puede perder tiempo cuando la granularidad de la información es demasiado alta o cuando se deseé una especificación muy minuciosa.
2. Si la especificación es demasiado vaga entonces tiende a convertirse en elemento de la ilación o educación.
3. Si la definición de los tipos de datos es estricta tiende a confundirse con la codificación de una parte del producto.
4. Si se encuentran demasiadas inconsistencias o ambigüedades en la especificación se pierde el tiempo en llevar a cabo las correspondientes correcciones en la etapa de ilación y educación.

6.6 Ejemplo de especificación de requisitos

A continuación, se proporciona un ejemplo de confección de las tablas de especificación obtenidas a partir de las tablas de ilación. Se continua con el ejemplo propuesto: Sistema de Inventarios. A pesar de contener varias tablas, estas han sido diseñadas para tener una lectura comprensible del problema a resolver. Puede notarse la asociación con las tablas de ilación, lo que permite lograr una adecuada trazabilidad.

Dentro del campo “Procedimiento” se encuentra una descripción en seudocódigo de la forma como se entiende la solución del problema. La minuciosidad del código va a depender del analista; si necesita codificación extrema puede hacer que en este campo se detallen líneas de código asociadas con un lenguaje de programación. Ello implica que el analista también debe de conocer aspectos de codificación en algún lenguaje de programación.

Asimismo, puede notarse la asociación que existe con los artefactos de diseño, los mismos que guardan absoluta relación con la codificación propuesta, así como de los autores, actores y fuentes. Algo que llama la atención es la propuesta de la grabación de imágenes que se consideran no recomendables cuando se trata de especificar requisitos.

Código especificación	ESP-0001
Nombre	Menú Principal.
Versión	01.01
Fecha	08/07/2020
Autor	AUT-0001, AUT-0002, AUT0004, AUT0008
Actor	ACT-0001, AUT-0002
Fuente	CON-0004
Código ilación	ILA-0001
Precondición	<ul style="list-style-type: none">• El desarrollador debe tener el repositorio maven.google.com declarado en el archivo build.gradle de nivel de módulo.• Tener importados los widgets correspondientes para Android import android.widget.• Contar con el documento Menu_principal.xml ya que este define la estructura de la vista.

Procedimiento	<p>Inicio</p> <ol style="list-style-type: none"> 1. Crear las variables de tipo "Button" para los botones de las distintas gestiones 2. private Button BUuarios, BCategorias, BProductos, BProveedores, BAlertas, BAbastecimiento, BSalidas, BReportes; 3. Instanciar la clase FirebaseAuth 4. Crear las variables de tipo "Button" para el botón Cerrar Sesión 5. private Button BCerrar_Sesion 6. Dentro de la función OnCreate() setear el content view, obtener la instancia activa en Firebase y referenciar las variables de tipo "Button" a sus respectivos botones. 7. OnCreate() 8. set content view Menu_principal.xml 9. mAuth = FirebaseAuth.getInstance() 10. BUuario = FindViewById(id.USuarios) 11. BCategorias = FindViewById(id.Categorias) 12. BProductos = FindViewById(id.Productos) 13. BProveedores = FindViewById(id.Proveedores) 14. BAlertas = FindViewById(id.Alertas) 15. BAbastecimiento = FindViewById(id.Abastecimiento) 16. BSalidas = FindViewById(id.Salidas) 17. BReportes = FindViewById(id.Reportes) 18. BCerrar_Cesion = FindViewById(id.Cerrar_Cesion) 19. Añadir OnClickListener a los Botones para que lance el Activity de Gestión correspondiente 20. BUuario.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Usuario.class))} 21. BCategoria.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Categoría.class))} 22. BProductos.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Productos.class))} 23. BProveedores.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Proveedores.class))} 24. BAlertas.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Alertas.class))} 25. BAbastecimiento.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Abstecimiento.class))} 26. BSalidas.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Salidas.class))} 27. BReportes.setOnClickListener(){Onclick();startActivity(Intent(Menu_Principal,Gestion_Reportes.class))} 28. Añadir OnClickListener al Botón Cerrar_Sesion para que cierre la sesión activa y nos devuelva al Login 29. BCerrar_Sesion.setOnClickListener(){Onclick()} 30. mAuth.signOut() 31. startActivity(Intent(Menu_Principal, Login.class)) 32. finish() 33. } <p>Fin</p>
Postcondición	Se visualiza la vista del menú principal correctamente con todos los componentes especificados y completamente funcionales.
Código de artefactos asociados	CON-0004: "Menú Principal"
Importancia	Vital
Estado	Pendiente
Comentario	Los componentes cargados en la vista son los nueve botones correspondientes a cada gestión: "Gestión de usuarios", "Categorías", "Productos", "Proveedores", "Gestión de alertas", "Abastecimiento", "Salidas", "Reportes", "Estadísticas" en sus respectivos contenedores.

Tabla 6.1: Requisito de especificación 0001

Código especificación	ESP-0002
Nombre	Vista "Gestión de Usuarios".
Versión	01.01
Fecha	14/07/2020
Autor	AUT-0002
Actor	ACT-0001, AUT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0001
Precondición	<ul style="list-style-type: none"> • Haber realizado correctamente los procedimientos de la ESP-0001 "Menú Principal". • Contar con el documento vista_gestionUsuarios.xml ya que este define la estructura de la vista. • El Usuario presiona el botón "Gestión de usuarios" de la ESP-0001: Vista "Menú Principal".
Procedimiento	<p>0 Inicio</p> <ol style="list-style-type: none"> 1. Se construye y muestra la vista "Gestión de usuarios" siguiendo la estructura de la CON-0003 "Pantalla General para cada módulo". Los elementos mostrados son: 2. Una ListView seleccionable con el id: "listViewUs" donde el administrador puede visualizar y seleccionar la lista de usuarios en el sistema. 3. Un Button "Registrar" con el id: "btnRegUs" donde el administrador puede registrar datos de un proveedor. 4. Un Button "Modificar" con el id: "btnModUs" donde el administrador puede modificar los datos de un usuario seleccionado. 5. Un Button "Salir" con el id: "btnExitUs" donde el administrador puede presionar para salir de la vista "Gestión de usuarios" y volver al "Menú Principal". 6. Dentro del Activity, se declaran los elementos mostrados en el formulario: 7. ListView listViewUs; Button btnRegUs, btnModUs y btnExitUs. 8. Dentro del Activity, en la función onCreate() referenciar los elementos mostrados y cargar el documento .xml 9. set content view vista_gestionUsuarios.xml 10. listViewUs = FindViewById(id.listViewUs) 11. btnRegUs = FindViewById(id.btnRegUs) 12. btnModUs = FindViewById(id.btnModUs) 13. btnExitUs = FindViewById(id.btnExitUs) 14. Dentro del activity, se añaden los OnClickListener correspondientes a cada botón para que tengan el comportamiento requerido. 15. Para realizar el proceso de "Registrar": btnRegUs.setOnClickListener() 16. Onclick({ "Iniciar ESP-0003: "Formulario de registro de usuario" }) 17. Para realizar el proceso de "Modificar": btnModUs.setOnClickListener() 18. { 19. Onclick({ "Iniciar ESP-0009: "Formulario de modificación de datos de usuario" }) 20. } 21. Para realizar el proceso de "Salir": btnExitUs.setOnClickListener() 22. { 23. Onclick({ "Iniciar ESP-0001: "Menú Principal" }) 24. Se ejecutan automáticamente los procedimientos de la ESP-0005: Listar usuario registrador en la base de datos 19 Fin
Postcondición	Se visualiza la gestión de usuarios correctamente con todos los componentes especificados y completamente funcionales.
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0003: "Pantalla general para cada módulo"
Importancia	Vital
Estado	Concluido

Comentarios	En esta especificación se construye la vista de la gestión de usuarios y las funcionalidades básicas, después de terminado los procesos, automáticamente se ejecuta la ESP-0005 donde se realiza el trabajo de obtener los proveedores usuarios en la tabla "Usuario".
-------------	--

Tabla 6.2: Requisito de especificación 0002

Código especificación	ESP-0003
Nombre	Formulario "Registrar datos de usuario".
Versión	01.01
Fecha	9/07/2020
Autor	AUT-0002
Actor	ACT-0001, AUT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0001
Precondición	<ul style="list-style-type: none"> • Haber realizado correctamente los procedimientos de la ESP-0001 "Menú Principal". • El Usuario presiona el botón "Registrar" de la ESP-0002: Vista "Gestión de Usuarios".
Procedimiento	<p>Inicio</p> <ol style="list-style-type: none"> 1. Se construye y muestra el formulario "Registrar datos de Usuario" siguiendo la estructura de la CON-0009 "Formularios de Registro de datos de usuario". Los elementos mostrados son: 2. Un EditText con el id: "nomUs" donde el administrador ingresa el "Nombre" del Usuario. 3. Un EditText con el id: "ApUs" donde el administrador ingresa los "Apellidos" del Usuario. 4. Un EditText con el id: "numContUs" donde el administrador ingresa el "Número de Contacto" del usuario. 5. Un EditText con el id: "dniRucUs" donde el administrador ingresa el "DNI/RUC" del usuario. 6. Un EditText con el id: "corrUs" donde el administrador ingresa el "Correo" del usuario. 7. Un EditText con el id: "dirUs" donde el administrador ingresa el "Dirección" del usuario. 8. Un Spinner con el id: "tipUs" donde el administrador selecciona el "Tipo de usuario" del usuario. Un EditText con el id: "ContraUs" donde el administrador ingresa la "Contraseña" del usuario. 9. Un RadioGroup con el id: "estUs" donde el administrador selecciona el "Estado" del usuario. 10. Un Button "Registrar" con el id: "btnRegUs" donde el administrador puede presionar una vez terminado el ingreso de datos. 11. Un Button "Cancelar" con el id: "btnCanRegUs" donde el administrador puede presionar para cancelar el registro de un usuario. 12. Dentro del Activity, se declaran los elementos mostrados en el formulario: EditText, spinner, RadioGroup y Button. 13. EditText edtNombre, edtAp, edtNumContacto, edtCoUs, edtDNI_RUC, edtCorreo y edtDir; un Spinner edtTipUs, un RadioGroup edtEstado y Button btnRegistrar y btnEliminar. 14. btnRegistrar = FindViewById(id.btnRegUs) 15. btnCancelar = FindViewById(id.btnCanRegUs) 16. El administrador ingresa por teclado los datos requeridos en el formulario. 17. El administrador selecciona de una lista de opciones los datos requeridos en el formulario. 18. El administrador selecciona de Radio buttons los datos requeridos en el formulario. 19. El administrador presiona en el botón "Guardar".

Procedimiento	<p>20. El método OnClick() del botón "Guardar" realiza el siguiente procedimiento:</p> <p>21. Se muestra un Alert Dialog con el mensaje "¿Desea registrar al Usuario?" con los botones "Aceptar" y "Cancelar".</p> <p>22. Si (Presiona Aceptar en el AlertDialog)</p> <p>23. El sistema recupera la información ingresada por el usuario en los campos del formulario.</p> <p>24. String Nombre = edtNombre.getText()</p> <p>25. String Apellidos = edtAp.getText()</p> <p>26. String numContacto = edtNumContacto.getText()</p> <p>27. String DNI_RUC = edtDNI_RUC.getText()</p> <p>28. String Correo = edtCorreo.getText()</p> <p>29. String dirección = edtDir.getText()</p> <p>30. String tipUs = edt.getSelectedItem().toString()</p> <p>31. String contraseña=edtCoUs().toString()</p> <p>32. String Estado = edtEstado.getText.toString()</p> <p>33. Se procede a realizar la ESP-0004 "Registrar un usuario en la base de datos".</p> <p>34. Sino</p> <p>35. Se cierra la alerta y el administrador es libre de modificar la información ingresada en el formulario.</p> <p>36. Finsi</p> <p>37. Si el administrador presiona en el botón "Cancelar" se cierra el formulario y se redirige al usuario a la vista "Gestión de Usuarios" (ESP-0002).</p> <p>Fin</p>
Postcondición	Redirigir al usuario a los procedimientos de la ESP-0004: "Registrar un usuario en la base de datos".
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0010: "Formularios de Registro de datos de usuario"
Importancia	Vital
Estado	Concluido
Comentarios	Se realiza la validación de la información recuperada del formulario en el .xml (Para ello se usa las expresiones regulares definidas en los comentarios de la ILA-0001: "Registrar datos de Usuario").

Tabla 6.3: Requisito de especificación 0003

Código especificación	ESP-0004
Nombre	Registrar un usuario en la base de datos.
Versión	01.02
Fecha	3/07/2020
Autor	AUT-0003, AUT-0006, AUT-0007
Actor	ACT-0001, AUT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0001
Precondición	<ul style="list-style-type: none"> • Haber realizado correctamente los procedimientos de la ESP-0003 "Formulario registrar datos de usuario". • Usuario presiona el botón "Guardar" de la CON-0009: "Formulario registrar datos de usuario".

Procedimiento	<p>Inicio</p> <ol style="list-style-type: none"> 1. Instanciar las clases FirebaseAuth y DatabaseReference 2. Usar el método createUserWithEmailAndPassword de FirebaseAuth. 3. Si (createUserWithEmailAndPassword == verdadero) entonces 4. Crear el Map Usuario. 5. Añadir campos a Map Usuario, por medio de put 6. usuario.put("Nombre",String); 7. usuario.put("Apellidos",String); 8. usuario.put("DNI", String); 9. usuario.put("Contraseña",String); 10. usuario.put("TipoUsuario",int); 11. usuario.put("Celular",String); 12. usuario.put("Correo Electrónico",String); 13. usuario.put("Direccion",String); 14. usuario.put("Estado",int); 15. Obtener la referencia del usuario activo en la variable String Id 16. Id = mAuth.getCurrentUser().Uid; 17. Añadir el map Usuario en el nodo ID con 18. Task<=> DatabaseReference.child("Usuario").child(id).setValue(Usuario) 19. //Validación del duplicado de DNI se define en la base de datos 20. Si (Task == verdadero) entonces 21. Mostrar un Toast con el mensaje "Usuario registrado correctamente". 22. Sino 23. Mostrar un Toast con el mensaje "Error al registrar usuario". 24. Finsi 25. Sino 26. Mostrar un Toast con el mensaje "Error al registrar usuario". 27. Finsi <p>Fin</p>
Postcondición	Registrar al usuario en la base de datos correctamente y mostrar un Toast con el mensaje "Usuario registrado correctamente".
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario" CON-0009: "Formularios de Registro de datos de usuario"
Importancia	Vital
Estado	Pendiente
Comentario	La validación de los campos se realizó en el archivo .xml Formulario "Registrar Usuario", de acuerdo a las validaciones en comentario de la ILA-0001 "Registrar datos de usuario".

Tabla 6.4: Requisito de especificación 0004

Código especificación	ESP-0005
Nombre	Listar usuarios registrados en la base de datos.
Versión	01.01
Fecha	14/07/2020
Autor	AUT-0002
Actor	ACT-0001, AUT-0002
Fuente	FUE-0004, FUE-0007
Código ilación	ILA-0001
Precondición	<ul style="list-style-type: none"> • El usuario ingresa a la vista "Gestión Usuarios" de la ESP-0016 y "Vista de Gestión de usuarios" de la ESP-0002. • Haber establecido la referencia a la base de datos Firebase DatabaseReference mRootReference mRootReference = FirebaseDatabase.getInstance().getReference()

Procedimiento	<p>Inicio</p> <ol style="list-style-type: none"> 1. Se construye el objeto correspondiente a los Usuarios con los mismos atributos que se definió en la tabla "Usuario" de la base de datos del sistema con sus getters y setters correspondientes. 2. Clase Usuario 3. Una variable de tipo int con el nombre "Key" 4. Una variable de tipo String con el nombre "Nombre" 5. Una variable de tipo String con el nombre "Apellidos" 6. Una variable de tipo int con el nombre "Celular" 7. Una variable de tipo String con el nombre "Correo" 8. Una variable de tipo String con el nombre "Dirección" 9. Una variable de tipo String con el nombre "Tipo_Usuario" 10. Una variable de tipo String con el nombre "Contraseña" 11. Una variable de tipo int con el nombre "DNI" 12. Una variable de tipo char con el nombre "Estado" 13. Constructor vacío del objeto Usuario 14. Getters y Setters correspondientes para cada variable. 15. Dentro del Activity correspondiente creamos un ArrayList para las entidades que recuperaremos de la Tabla "Usuario" de la base de datos del sistema. 16. ArrayList <Usuario> UsuarioList 17. Añadimos un eventListener al nodo "Usuario" dentro de la base de datos del sistema 18. mRootReference.child("Usuario").addEventListenner() 19. INICIO EVENT LISTENER I 20. Dentro de la función OnDataChange(DataSnapshot) creamos un bucle de tipo for para recorrer todos los nodos anidados en el snapshot. 21. OnDataChange(DataSnapshot){ 22. INICIO FOR DataSnapshot c: DataSnapshot.getChildren() 23. Añadimos un eventListener al nodo "Usuario.Key" dentro de la base de datos del sistema. 24. mRootReference.child("Usuario").child(c.getKey).addEventListenner() 25. INICIO EVENT LISTENER II 26. Dentro de la función OnDataChange(DataSnapshot) creamos el Objeto Usuario con los datos que recuperemos de la base de datos del sistema y las insertamos en el ArrayList que hemos creado para dichas entidades. 27. OnDataChange(DataSnapshot){ 28. Usuario user = DataSnapshot.getValue(Usuario.class) 29. user.SetKey(c.getKey) 30. UsuarioList.add(user) 31. } 32. } FIN EVENT LISTENER II 33. Fin for 34. } FIN EVENT LISTENER I <p>Fin</p>
Postcondición	ArrayList con todos los registros de la tabla "Usuario" de la base de datos del sistema.
Código de artefactos asociados	DBD: "Diccionario de la base de datos", Tabla: "Usuario"
Importancia	Vital
Estado	Concluido
Comentarios	Ninguno

Tabla 6.5: Requisito de especificación 0005

De manera similar se logra plasmar las 99 especificaciones asociadas con las correspondientes ilaciones.

CAPÍTULO 7

Trazabilidad de Requisitos

7.1 Introducción

El desarrollo de software puede ser llevado a cabo de manera tradicional o respetando aspectos de calidad en su construcción. Normalmente no son aplicados estos aspectos y se construye siguiendo criterios personales producto de la experiencia o porque la intuición del analista lo dice. Desde este ángulo de vista, el desarrollo se torna como una actividad con la sola finalidad de que haga lo que el cliente solicita.

En la propia construcción del software surgen artefactos que deben ser empleados para representar elementos estáticos y dinámicos del sistema producto del lenguaje de modelado empleado. En muchas oportunidades estos artefactos son solo empleados con criterios personales sin tomar en cuenta los artefactos diseñados y las explicaciones del cliente; producto de ello se genera un desorden en su concepción.

Bajo este criterio de construcción, cuando se trata de encontrar errores en los artefactos se desconoce de dónde provienen o a cuáles otros artefactos afectan. Solo se busca el error en el artefacto donde se encuentra el mismo sin hacer un análisis de sus consecuencias; es por esto que es importante encontrar formas de como conocer el entorno total y su afectación a los artefactos diseñados.

Es por ello que la trazabilidad es de suma importancia. Trazar o llevar la ruta de afectación de todos los artefactos empleados es importante incluido el catálogo de requisitos; y es precisamente el catálogo de requisitos el artefacto donde se deben de almacenar todos los elementos a ser trazados cuando se trata de corregir errores introducidos en la construcción del software. Esta condicionalidad no altera la estructura del catálogo, pero si mantiene la línea secuencial desde donde aparecen los artefactos.

Según la guía del Business Analysis Body of Knowledge (BABOK), define la trazabilidad como: *"La trazabilidad de requerimientos es la capacidad de registrar las relaciones existentes entre*

la necesidad dada por un interesado, usuario, cliente o por el stakeholder, el requerimiento de proyecto y la solución implementada finalmente". Esta definición es respaldada por muchos investigadores y es aplicada en una multiplicidad de trabajos relacionados con la construcción del software. El tema es importante ya que permite agregar orden cuando se trata de llevar a cabo el mantenimiento de los requisitos y de sus artefactos inherentes de tal forma que se puede organizar el historial de cambios en los artefactos incluyendo la aparición de nuevos de ellos.

La trazabilidad ayuda a determinar el camino donde se reflejan las necesidades del cliente y su representación en el software; es por eso que la trazabilidad de requisitos y requerimientos son la base para el control del alcance, riesgo, cronograma, costo y comunicaciones a lo largo del proyecto, así como de la forma como van cambiando los requisitos a lo largo del mismo. Se debe tener presente que los requisitos no funcionales también afectan a los requisitos funcionales.

7.2 Elementos de la trazabilidad

La trazabilidad contiene un conjunto de elementos que son esenciales para su ejecución y/o concepción. A continuación, se tratan algunos de ellos:

1. Conjunto de objetivos de la organización: Los objetivos de la organización deben estar redactados de forma clara con la finalidad de entender lo que se desea construir permitiendo un buen desenlace en la construcción del software y logrando lo que se espera con respecto a ello.
2. El artefacto de educación de requisitos: Este conjunto de tablas debe resolver el problema de los módulos que integrarán el software como solución genérica de las necesidades del cliente y como una primera visión del producto. El refinamiento del contenido de estas tablas implica una primera visión de la mejora continua.
3. El artefacto de ilación de requisitos: Este documento explica lo que el analista entiende como modelo del negocio y como enfoca su solución como parte de la construcción del software. La ilación permite refinar los procesos del negocio y su asociación con la solución a presentar desde el ángulo de vista de la automatización de los procesos de la organización.
4. El artefacto de especificación de requisitos: Conjunto de plantillas orientadas a la especificación de la codificación del producto de software y en donde se plasman aspectos relacionados con el lenguaje de programación como lo son los tipos de datos, estructuras de datos, estructuras de datos espaciales, interfaces internas entre otras.
5. Conjunto de artefactos definidos según el lenguaje de modelado empleado: Dependiendo del lenguaje de modelado a adoptar, se deben de definir cuáles son los artefactos que se van a emplear para resolver el problema en cuestión. La selección también depende de los artefactos escogidos y de la complejidad del problema a resolver.

6. Conjunto de criterios para la corrección de inconsistencias y ambigüedades: Se deben preparar los procedimientos para resolver los problemas de inconsistencias y ambigüedades que se pudieran presentar en los requisitos o en los artefactos relacionados. Existen requisitos que por su naturaleza o especialidad necesitarán un procedimiento especial.
7. Conjunto de criterios para el control de versiones y la mejora continua: Una vez resueltas las inconsistencias y ambigüedades entre los artefactos definidos, se deben adoptar criterios para adaptar un control de versiones que vaya sosteniendo el historial de cambios efectuados incluidos los cambios a los diagramas definidos.
8. Conjunto de relaciones: Estas relaciones permiten entender cómo se asocian los artefactos tanto de manera horizontal como vertical; permitiendo relacionar cambios asociados a cada uno de ellos. Estos cambios deben ser documentados porque permite, en un futuro, otorgarle la sostenibilidad al producto de software.

7.3 Proceso de trazabilidad

El proceso de trazabilidad debe llevarse a cabo con todos los artefactos generados para el análisis y diseño del software, incluido su correspondiente codificación. La figura 7.1 indica que en primer lugar se debe de hacer un análisis de las tablas de educación, al encontrar inconsistencias y/o ambigüedades estas deben ser resueltas, mediante procedimiento, haciendo un análisis vertical, o sea entre tablas de educación hasta definir los módulos que serán considerados dentro del producto.

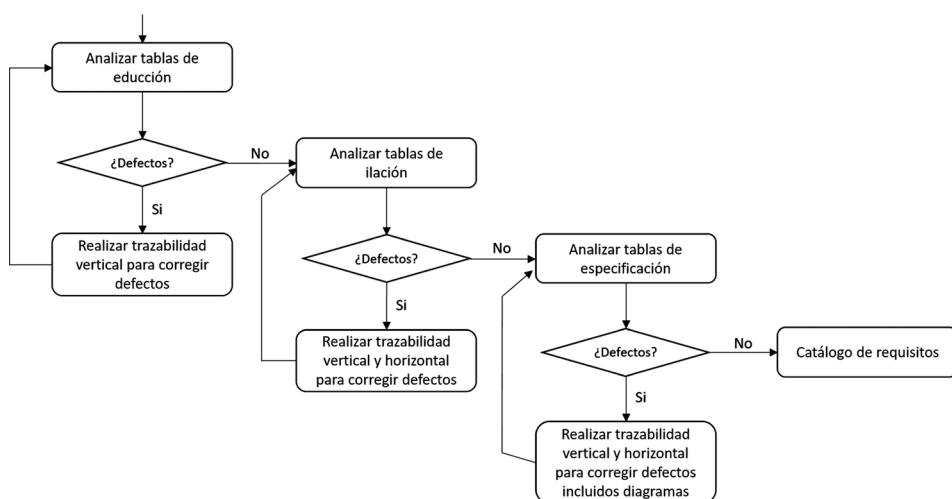


Figura 7.1: Proceso de trazabilidad de requisitos

Corregidas las tablas de educación, se proceden de igual forma con las tablas de ilación haciendo un correctivo tanto horizontal como vertical; es decir entre tablas de ilación y su correspondiente relación con las tablas de educación. Hechas las correcciones se deben

versionar las correspondientes tablas para mostrar la mejora continua del proceso.

Finalmente, se realiza el análisis de las tablas de especificación solucionando los defectos encontrados bajo el ángulo de vista de la trazabilidad vertical y horizontal. La trazabilidad horizontal se logra al navegar por las asociaciones entre las tablas de ilación y de educación y la vertical se logra por la asociación de las tablas de especificación con los diagramas del lenguaje de modelado empleado.

Todos los diagramas diseñados como los son: clases, secuencias, interacción, colaboración, objetos, procesos e incluso las interfaces gráficas de usuario deben ser codificados e incrustados dentro de las tablas de especificación o ilación. Cuando se corrijan los defectos, estos se vislumbran en las tablas y en los diagramas.

En general, estas codificaciones pueden ser agregadas en las tablas de educación, ilación o especificación o en las tablas que el analista lo viere por conveniente. Cuando se encuentre un defecto entonces se procede a realizar las correcciones, y obviamente a versionar, en todos los artefactos asociados con la tabla analizada.

7.4 Trazabilidad simple

Una trazabilidad simple consiste en resolver el problema de las inconsistencias y/o ambigüedades entre un par de tablas del mismo nivel o de diferente nivel. Por ejemplo, encontradas las ambigüedades en una tabla de ilación se asocia con la tabla de educación y los defectos se corrigen en cascada generando el control de versiones correspondiente. La tabla 7.1 muestra esta característica.

Atributos	Descripción
Código ilación	ILA-0004
Versión	1.0
Autor	...
Fuente	...
Dependencia	EDU-0002
Descripción	...
Importancia	...
Estado	...
Comentarios	...

Tabla 7.1: Trazabilidad simple

La tabla de ilación 4 tiene una asociación con la tabla de educación 2. Se corrigen las inconsistencias o ambigüedades de la tabla ILA-0004 pero se reportan a la tabla EDU-0002

donde también se procede a su corrección para finalmente generar una nueva versión de las mismas (tabla 7.2). Las versiones anteriores son guardadas y pasan a conformar el historial del catálogo de requisitos.

Atributos	Descripción
Código ilación	ILA-0004
Versión	3.0
Autor	...
Fuente	...
Dependencia	EDU-0002
Descripción	...
Importancia	...
Estado	...
Comentarios	...

Tabla 7.2: Trazabilidad simple iterada

Se debe tener presente que existe la posibilidad de que al corregir una tabla de ilación, y consecuentemente la tabla de educación, es probable que se pueda generar una multiplicidad de tablas producto de lograr la atomicidad de criterios o conceptos. Estas nuevas tablas son incrustadas respetando la codificación correspondiente y las demás tablas son recodificadas y nuevamente asociadas con sus tablas antecesoras. Se debe recordar que también los otros artefactos asociados son recodificados para cumplir con la misma filosofía.

7.5 Trazabilidad múltiple

Una trazabilidad múltiple es aquella en la cual las correcciones afectan al conjunto de artefactos asociados. Todos los artefactos asociados son analizados y corregidos y producto de ello pueden nacer otro conjunto de artefactos que deben ser codificados e incrustados en el catálogo de requisitos. La figura 7.2 muestra los niveles de asociación.

Absolutamente todo artefacto que se emplea para la construcción de un producto de software debe figurar en el catálogo de requisitos y asociarse con las tablas de educación, ilación y especificación. También lo debe hacer las entrevistas y el código así como todos los diagramas empleados para tal finalidad.

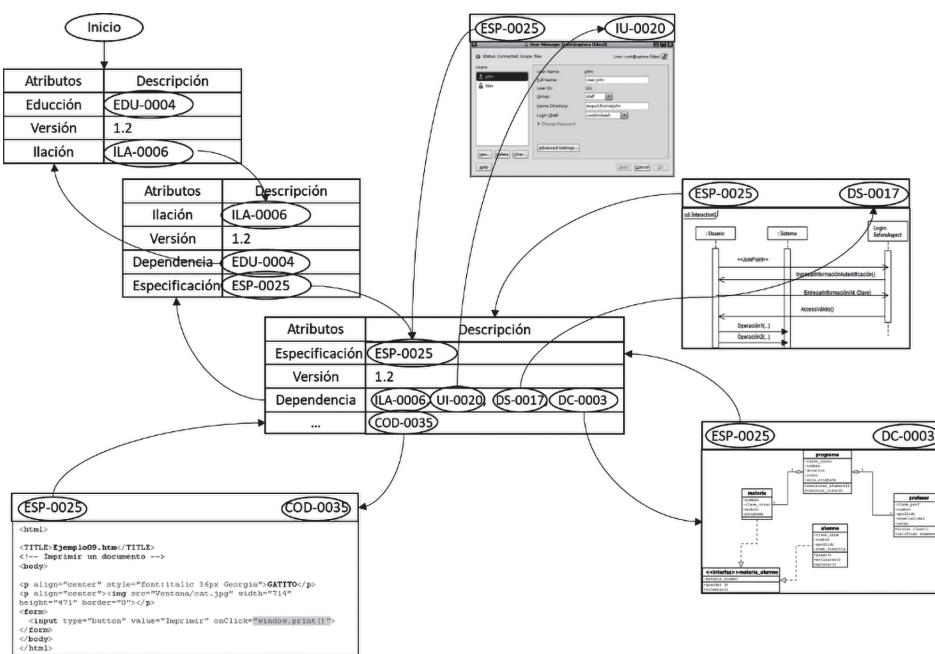


Figura 7.2: Trazabilidad múltiple

7.6 Implementación de la trazabilidad

Para implementar la trazabilidad simple y múltiple, lo cual implica su versión vertical y horizontal, se debe seguir el siguiente procedimiento:

1. Codificar cada tabla de educación agregando el prefijo EDU-000X. Este prefijo será empleado en las tablas de la siguiente etapa como medida de asociación entre los artefactos pertenecientes al proceso.
2. Codificar cada tabla de ilación y en sus dependencias agregar el código de las tablas de educación asociadas. Se agrega el prefijo ILA-000Y con la finalidad de indicar la respectiva asociación.
3. Codificar cada tabla de especificación y en sus dependencias agregar el código de cada tabla de ilación asociada. Agregar el prefijo ESP-000Z el mismo que simbolizará la asociación con los otros artefactos.
4. Codificar cada artefacto de diseño e incrustarlo, como dependencia, dentro de las tablas de ilación o especificación según fuere el caso. Agregar los prefijos correspondientes como: DS-000A, IU-000B, DS-000C, DC-000D entre otros.

La tabla 7.2 muestra un ejemplo de su implementación y se puede notar claramente que seguir el rastro de los artefactos asociados son fáciles de hacerlo y también de corregirlos por el ahorro de tiempo en su detección.

7.7 Análisis de la trazabilidad

La trazabilidad en la etapa de educación se hace de manera vertical lo que implica que se debe de buscar la asociación entre conceptos propios del sistema a construir. En esta primera etapa se busca definir los módulos que conforman el producto a construir y su correspondencia con el modelo del negocio de la organización.

En la etapa de ilación, la trazabilidad de lleva a cabo de manera vertical y horizontal lo que implica un análisis entre las tablas de ilación y educación y de los artefactos de diseño asociados con estas tablas. También se lleva a cabo un análisis entre tablas de ilación ya que es posible una dependencia producto del mal concepto del diseño.

En la etapa de especificación, la trazabilidad se hace de manera vertical u horizontal. Se asocian las tablas de especificación con las de ilación y educación o entre tablas de especificación. También se asocian los artefactos de diseño. Todos estos elementos son versionados de manera adecuada o siguiendo un procedimiento previsto por el analista.

Encontradas y corregidas las inconsistencias y ambigüedades se procede a realizar un análisis vertical y horizontal para detectar las tablas o artefactos afectados. Si se encuentran elementos afectados se procede a corregirlos versionando los mismos. La solución es una burbuja en el que al final todos los elementos asociados deben estar corregidos.

7.8 Ejemplo de trazabilidad

Aplicaremos la noción de trazabilidad sobre el sistema visto a lo largo del texto. En vista de que el sistema contiene una gran cantidad de tablas sólo se facilitarán aquellas que demuestran la trazabilidad dentro de este contexto, incluyendo los artefactos de diseño denominado interfaz gráfica de usuario.

Por ejemplo, la tabla 4.1 que corresponde a la tabla de educación 0001 guarda una correspondencia con las tablas de ilación ILA-0001, ILA-0002, ILA-0003 e ILA-0004. La tabla de ilación ILA-0001 mantiene una relación con las tablas de especificación ESP-0001, ESP-0002, ESP-0003, ESP-0004 y ESP-0005 mientras que la tabla de ilación ILA-0002 mantiene relación con las tablas de especificación ESP-0006 y ESP-0007; por otro lado, la tabla de ilación ILA-0003 guarda relación con las tablas de especificación ESP-0008, ESP-0009 y ESP-0010. Finalmente, la tabla de ilación ILA-0004 mantiene la relación con las tablas de especificación ESP-0011 y ESP-0012.

Un detalle importante es que la tabla de ilación ILA-0003 (tabla 5.3) presenta relación con otro artefacto asociado como es el diccionario de la base de datos y el formulario en general codificado como CON-0010 (figura 7.4); asimismo, la tabla de especificación ESP-0001(tabla 6.1) tiene relación con la interfaz gráfica de usuario "Menú Principal" codificado como CON-0004 (figura 7.3).

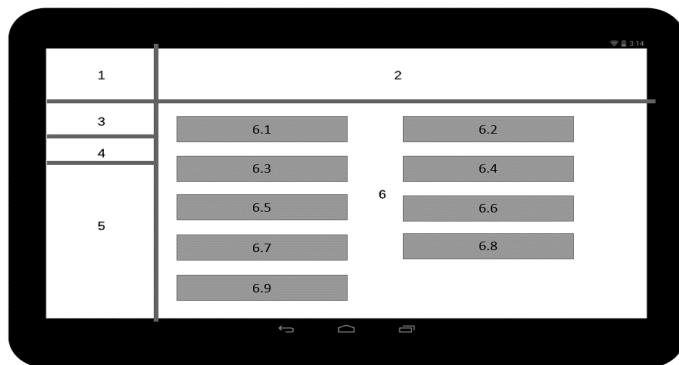


Figura 7.3: Interfaz CON-0004

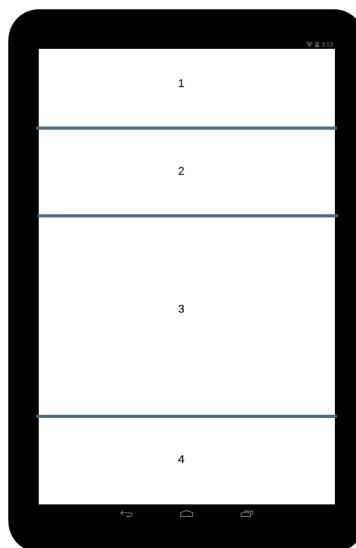


Figura 7.4: Interfaz CON-0010

El esquema final de trazabilidad para el producto de software “Sistema de Inventarios de la Bodega ELVITA”, que se propone como ejemplo, se muestra en la tabla 7.3.

EDUCCIÓN	ILACIÓN	ESPECIFICACIÓN	ARTEFACTOS ASOCIADOS
EDU-0001	ILA-0001	ESP-0001	CON-0004
		ESP-0002	CON-0003, DBD
		ESP-0003	CON-0010, DBD
		ESP-0004	CON-0009, DBD
		ESP-0005	DBD
	ILA-0002	ESP-0006	DBD, CON-0001, CON-0002
		ESP-0007	DBD, CON-0001, CON-0002
	ILA-0003	ESP-0008	DBD
		ESP-0009	DBD, CON-0010
		ESP-0010	DBD, CON-0010
	ILA-0004	ESP-0011	DBD, CON-0010
		ESP-0012	DBD, CON-0001, CON-0002
EDU-0002	ILA-0005	ESP-0013	DBD
		ESP-0014	Ninguna
		ESP-0015	Ninguna
		ESP-0016	CON-0003
	ILA-0006	ESP-0017	DBD, CON-0003
	ILA-0007	ESP-0018	DBD, CON-0010
		ESP-0019	DBD
	ILA-0008	ESP-0020	DBD
	ILA-0009	ESP-0021	DBD
EDU-0003	ILA-0010	ESP-0022	DBD, CON-0010
		ESP-0023	DBD, CON-0010
		ESP-0024	DBD, CON-0003
	ILA-0011	ESP-0025	DBD, CON-0010
		ESP-0026	DBD, CON-0010
	ILA-0012	ESP-0027	DBD, CON-0010
		ESP-0028	DBD, CON-0010
	ILA-0013	ESP-0029	CON-0012
		ESP-0030	DBD, CON-0010
		ESP-0031	DBD, CON-0010
	ILA-0014	ESP-0032	DBD, CON-0010
		ESP-0033	DBD, CON-0010
	ILA-0015	ESP-0034	DBD, CON-0010
		ESP-0035	DBD, CON-0010
		ESP-0036	DBD

EDU-0004	ILA-0016	ESP-0037	DBD, CON-0003
		ESP-0038	DBD
	ILA-0017	ESP-0039	DBD, CON-0010
		ESP-0040	DBD, CON-0010
	ILA-0018	ESP-0041	DBD, CON-0010
		ESP-0042	DBD, CON-0010
	ILA-0019	ESP-0043	DBD
		ESP-0044	DBD, CON-0003
	ILA-0020	ESP-0045	DBD, CON-0010
		ESP-0046	DBD, CON-0010
EDU-0005	ILA-0022	ESP-0047	DBD
		ESP-0048	DBD
		ESP-0049	DBD, CON-0003
		ESP-0050	DBD
		ESP-0051	CON-0006
		ESP-0052	DBD, CON-0010
		ESP-0053	DBD, CON-0010
	ILA-0023	ESP-0054	DBD, CON-0010
		ESP-0055	DBD
		ESP-0056	DBD, CON-0010
EDU-0006	ILA-0024	ESP-0057	DBD
	ILA-0025	ESP-0058	DBD, CON-0003
	ILA-0026	ESP-0059	DBD
	ILA-0027	ESP-0060	DBD, CON-0003
		ESP-0061	DBD
		ESP-0062	DBD, CON-0003
	ILA-0028	ESP-0063	DBD
		ESP-0064	DBD, CON-0003
	ILA-0029	ESP-0065	DBD, CON-0010
		ESP-0066	DBD, CON-0003
EDU-0007	ILA-0030	ESP-0067	DBD, CON-0003
	ILA-0031	ESP-0068	DBD, CON-0010
	ILA-0032	ESP-0069	DBD, CON-0010
	ILA-0033	ESP-0070	DBD, CON-0003
		ESP-0071	DBD, CON-0010
		ESP-0072	DBD, CON-0010
	ILA-0034	ESP-0073	DBD
		ESP-0074	DBD, CON-0003
	ILA-0035	ESP-0075	DBD, CON-0010
		ESP-0076	DBD, CON-0010
	ILA-0036	ESP-0077	DBD, CON-0010

EDU-0008	ILA-0037	ESP-0078	CON-0003	
		ESP-0079	CON-0010	
		ESP-0080	DBD	
		ESP-0081	DBD	
	ILA-0038	ESP-0082	CON-0010	
		ESP-0083	DBD	
		ESP-0084	DBD	
	ILA-0039	ESP-0085	DBD, CON-0003, CON-0004	
	ILA-0040	ESP-0086	DBD, CON-0011	
	ILA-0041	ESP-0087	CON-0003	
EDU-0009		ESP-0088	CON-0003	
		ESP-0089	DBD	
		ESP-0090	DBD	
ILA-0042	ESP-0091	CON-0003		
	ESP-0092	DBD		
	ESP-0093	DBD, CON-0008		
ILA-0043	ESP-0094	CON-0003		
	ESP-0095	DBD		
	ESP-0096	DBD		
ILA-0044	ESP-0097	CON-0003		
	ESP-0098	DBD, CON-0003		
ILA-0045	ESP-0099	CON-0011		

Tabla 7.3: Esquema final de trazabilidad

CAPÍTULO 8

Requisitos no Funcionales

8.1 Introducción

Un tema que merece especial atención es el de los Requisitos No Funcionales (RNF). Este artefacto, comúnmente no es diseñado ya que no se ve involucrado con las funcionalidades del sistema a pesar de que afecta severamente en los requisitos funcionales. En oportunidades se diseñan los requisitos funcionales sin considerar las características de los requisitos no funcionales, dando como resultado una pérdida de tiempo y costo cuando se intenta arreglar el defecto.

Poort Eltjo R. menciona que la falta de conocimiento sobre la relación entre los requerimientos no funcionales y las soluciones arquitectónicas a menudo conduce a problemas en proyectos de la vida real. El artículo presenta un modelo que se concentra en la asignación de requerimientos no funcionales en requerimientos funcionales para el diseño de arquitectura. Crea un marco que proporciona un modelo y un método repetible para transformar los requerimientos en conflicto en una descomposición del sistema [39].

El artículo presenta el marco y analiza dos casos en los que se aplica el método. En un caso, el método se utiliza con éxito para reconstruir la estructura de alto nivel de un sistema a partir de sus requerimientos. El segundo caso es uno en el que el método se utilizó realmente para crear un diseño de sistema que se ajuste a las necesidades de las partes interesadas, y que es reproducible a partir de sus requerimientos [39].

Sadana Vishal et. al. indican que los conflictos entre los requerimientos no funcionales a menudo se identifican subjetivamente y no existe un análisis de conflictos en la práctica. Los enfoques actuales no logran capturar la naturaleza de los conflictos entre los requerimientos no funcionales, lo que dificulta la tarea de resolución de conflictos. Su investigación proporciona un marco para el análisis de conflictos entre requerimientos no funcionales utilizando el análisis integrado de requerimientos funcionales y no funcionales [40].

El marco identifica y analiza conflictos basados en relaciones entre atributos de calidad, funcionalidades y restricciones. Dado que las declaraciones de requerimientos mal estructurados suelen dar lugar a especificaciones confusas; también desarrolla formas canónicas para representar requerimientos no funcionales. El resultado es una jerarquía de conflictos que refina los conflictos entre requerimientos no funcionales nivel por nivel [40].

Abdul H. et. al. mencionan que la identificación de conflictos entre requerimientos no funcionales a menudo se identifica intuitivamente, lo que perjudica las prácticas de análisis de conflictos. La investigación propone un nuevo modelo para identificar conflictos entre requerimientos no funcionales. El modelo propuesto utiliza el mecanismo matricial para identificar los conflictos basados en la calidad entre los requerimientos no funcionales [41].

Los conflictos potenciales se identifican mediante el mapeo de atributos de calidad conflictivos de bajo nivel a funcionalidades de bajo nivel utilizando las matrices. El modelo propuesto logra la identificación de conflictos entre los requerimientos de productos y procesos, identifica conflictos falsos, disminuye la sobrecarga de documentación y mantiene la transparencia de los conflictos identificados [41].

Fu Yun, Li Minqiang y Chen Fuzan indican que el cambio de requerimientos es una fuente importante de riesgo para los proyectos de desarrollo de software. La predicción de los cambios en los requerimientos plantea un desafío en la gestión de riesgos de software, especialmente en las primeras etapas de los proyectos de desarrollo de software. La investigación predice el riesgo de propagación del cambio en términos de probabilidad de propagación del cambio e impacto del cambio [42].

Primero, se discute el proceso de cambios en los requerimientos de software. Luego, se establece un modelo probabilístico basado en la matriz de estructura de diseño para evaluar el riesgo de propagación del cambio desde los requerimientos hasta la arquitectura del software. Además, el modelo propuesto se utiliza para estimar el cronograma y el costo de un proyecto de desarrollo de software [42].

Bo et. al. indican que los requerimientos de software, y en especial los requerimientos no funcionales, son considerados como una condición imprescindible para la producción de software de alta calidad. Como es ampliamente aceptado, el modelamiento de objetivos no funcionales como son los marcos de trabajo NFR (del inglés, Non Functional Requirements), por lo general emplea el modelo de árbol, y presenta un proceso interactivo para el análisis de requerimientos no funcionales. Sin embargo, aún existen algunos problemas durante la identificación de estado [43].

El artículo se basa en el conocido modelo de modales objetivo de razonamiento NFR, que distingue a la suposición de mundo cerrado y el mundo abierto, y propone un mecanismo automático para el razonamiento de metamodelos NFR con el fin de identificar los estados que satisfacen las raíces de los árboles objetivo según la contribución que hacen las hojas. Son transformados en estados que satisfacen los objetivos de sus padres. Entonces los

padres satisfacen los estados que se desprenderán de acuerdo con el razonamiento de reglas derivadas de diferentes relaciones de descomposición [43].

Supakkul et. al. mencionan que los requerimientos no funcionales (NFR), como la seguridad y el costo, son generalmente subjetivos y muchas veces sinérgico o conflictivas entre sí. Tratar adecuadamente tales NFR requiere una gran cantidad de conocimientos. Sin embargo, existen algunos patrones para hacer frente a este tipo de conocimiento de NFR. En este trabajo, presentan cuatro tipos de patrones para capturar y reutilizar los NFR - patrón objetivo, patrón de problemas, patrón de alternativas y el patrón de selección. Los patrones NFR pueden ser representados visualmente, y organizados bajo una normativa de especialización para así construir patrones más grandes [44].

Chin-Lun Liu menciona que el analizar los conflictos en los requerimientos no funcionales es una tarea importante en grandes proyectos de desarrollo de sistemas de software. Muchos de los requerimientos no funcionales que se acumulan con el tiempo pueden variar. Los analistas de sistemas suelen mantener los requerimientos no funcionales de forma incremental. Los problemas de sobrecarga de información y la rotación de los empleados pueden complicar la detección de conflictos en el proceso de evolución del requisito no funcional. Este trabajo propone un detector de conflictos en la evolución del requisito no funcional y utiliza ontologías como fundamento teórico para la detección automática de los conflictos [45].

Rao indica que la identificación de los requerimientos no funcionales es importante para el éxito del desarrollo y despliegue del producto de software. La aceptación del producto de software por parte del cliente depende de los requerimientos no funcionales que se incorporan en el software. Para ello, es necesario identificar todos los requerimientos no funcionales necesarios por todas las partes interesadas. En la literatura muchos enfoques no están disponibles para este propósito [46].

Por lo tanto, propone un enfoque de cuatro capas de análisis para la identificación de los requerimientos no funcionales. El enfoque por capas propuesta tiene muchas ventajas sobre el enfoque que no está en capas. Como parte de este enfoque proponen algunas reglas para ser usado en cada capa. El enfoque se aplica con éxito en dos estudios. Los requerimientos no funcionales identificados se validaron utilizando una lista de verificación y, además, la conformidad de los requerimientos no funcionales identificados se calcula utilizando una métrica [46].

Arda Goknila et. al. indican que, siguiendo la evolución de las necesidades del negocio, los requerimientos de los sistemas de software cambian continuamente y surgen nuevos requerimientos con frecuencia. Los documentos de requerimientos son a menudo artefactos textuales con una estructura que no se da explícitamente. Cuando se introduce un cambio en un documento de requerimientos, el analista puede tener que analizar manualmente todos los requerimientos para un solo cambio. Esto puede dar, como resultado, descuidar el impacto real de un cambio. En consecuencia, el costo de implementar un cambio puede ser varias veces mayor de lo esperado [47].

8.2 Definición

De acuerdo con Thayer, "...en ingeniería de software, un requisito no funcional es un requisito de software que describe no lo que hará el software, sino como lo hará, por ejemplo, los requisitos de rendimiento del software, los requisitos de la interfaz externa del software, las restricciones de diseño del software y los atributos de calidad del software. Los requisitos no funcionales son difíciles de probar; por lo tanto, generalmente se evalúan subjetivamente..." [48].

Sommerville hace la siguiente definición: "Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema." [9].

María Gómez menciona lo siguiente: "Los requerimientos no funcionales describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema. Restringen los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, el tipo de proceso de desarrollo a utilizar, fiabilidad, tiempo de respuesta, capacidad de almacenamiento. Los requerimientos no funcionales ponen límites y restricciones al sistema." [49].

María Ventura define lo siguiente: "Los requerimientos no funcionales describen únicamente atributos del sistema o atributos del ambiente del sistema y pueden ser, por ejemplo: requerimientos de interfaz, de diseño, de implementación, legales, físicos, de costo, de tiempo, de calidad, de seguridad, de construcción, de operación, entre otros." [50].

En conclusión, se puede decir que los requisitos no funcionales son requisitos que afectan o parametrizan a la funcionalidad del sistema restringiendo tareas en el contexto general del sistema.

8.3 Clasificación

Los requisitos no funcionales presentan interesantes clasificaciones. Sommerville [9] clasifica a estos en tres tipos: requisitos no funcionales orientados al producto, requisitos no funcionales orientados a la organización y requisitos no funcionales externos los mismos que se encuentran subdivididos en interoperabilidad, éticos y legislativos.

Los requisitos de eficiencia los subdivide en requisitos de rendimiento y de espacio, mientras que los organizacionales son subdivididos en requisitos de entrega, de implementación y estándares. Los requisitos legislativos los subdivide en requisitos de privacidad y de seguridad. La figura 8.1 muestra su estructura.

En su novena edición reorganiza esta clasificación y propone una estructura nueva en la que mantiene los requisitos orientados al producto, organización y externos, pero subdivide los

requisitos del producto en requisitos de usabilidad, eficiencia, dependibilidad y seguridad, mientras que los requisitos orientados a la organización los clasifica en requisitos de entorno, organizacionales y de desarrollo. Asimismo, los requisitos externos los cataloga en requisitos regulatorios, éticos y legislativos. La figura 8.2 muestra esta nueva estructura.

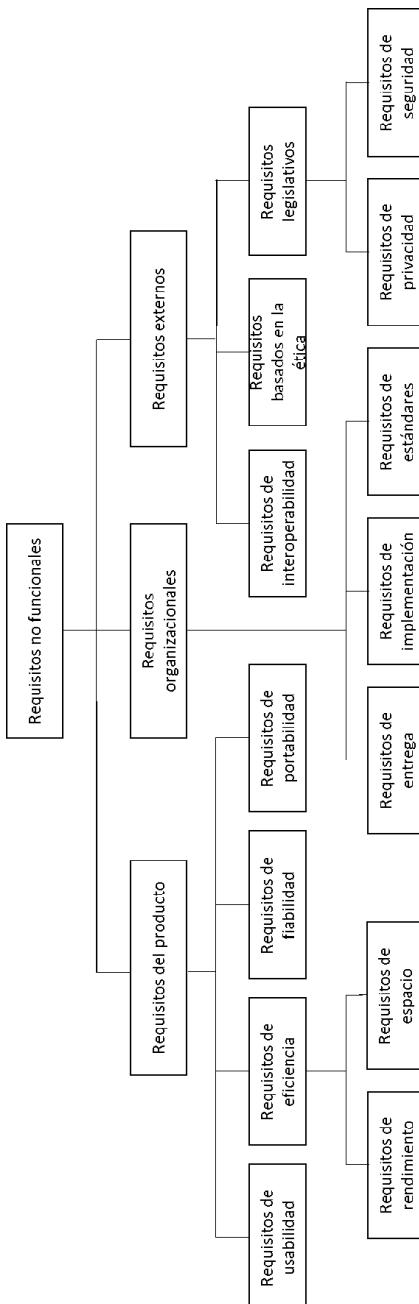


Figura 8.1: Clasificación de los Requisitos No Funcionales según Sommerville. Fuente [51]

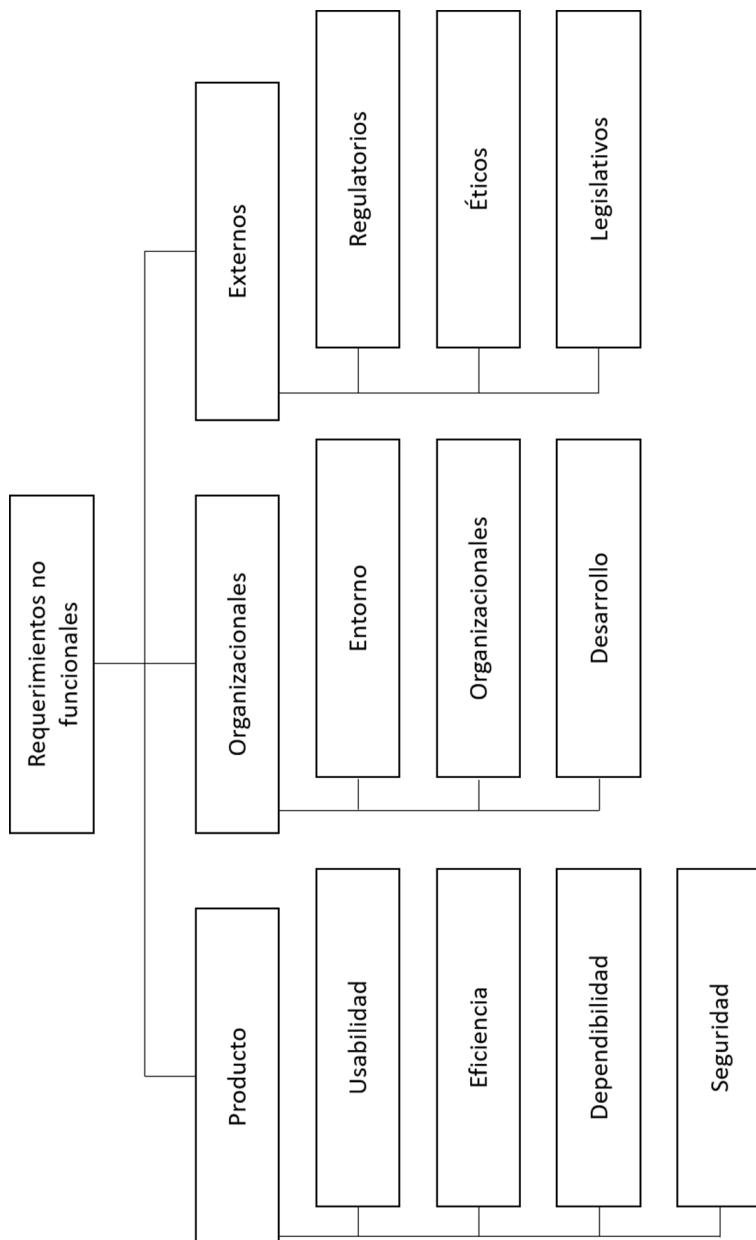


Figura 8.2: Clasificación de los Requisitos No Funcionales según Sommerville. Fuente [9]

8.4 Efecto de los requisitos no funcionales

Normalmente los requisitos no funcionales no son contemplados en la práctica u otras veces son capturados y definidos al final de la obtención de los requisitos funcionales. Esta práctica es común debido al poco interés que se pone a este tipo de artefactos ya que los desarrolladores se preocupan más de explicitar las funcionalidades del sistema sin tomar en cuenta los efectos colaterales que pueden acarrear los requisitos no funcionales.

¿Por qué este desentendimiento? Los Ingenieros de Software suelen pensar que los requisitos no funcionales no aportan nada en la construcción del software y que solo están ahí para complementar la documentación del producto; incluso suelen elaborar catálogos de requisitos sin considerar a los requisitos no funcionales y mucho menos relacionarlos con los requisitos funcionales.

Desde nuestro punto de vista, se debe mantener el respeto por ambos tipos de requisitos y consideramos que primeramente deben ser definidos los requisitos no funcionales para tener un amplio espectro de las restricciones que se introducirán en los requisitos funcionales y así poder obtener un catálogo de requisitos libre de inconsistencias y ambigüedades y con una trazabilidad que permita resolver los inconvenientes con costos bajos.

Veamos, que sucede si al cliente no le hacemos la pregunta: ¿Cuenta usted con internet en su negocio? y luego nos ponemos a definir las funcionalidades del producto; finalmente logramos construirlo e instalarlo en los dispositivos del cliente; ante la presentación del producto el cliente nos hace la pregunta: ¿Qué pasa si el internet no funciona?, ¿cómo funciona el sistema?. Sólo estas interrogantes nos hacen dar cuenta del error cometido y de la importancia de los requisitos no funcionales.

Pero, ¿De quién es la culpa?, ¿del cliente o de los analistas?. Obviamente de los analistas que no han sabido tomar en consideración los requisitos no funcionales y se han preocupado solo de la funcionalidad del sistema. Después de ello, los analistas tienen que perder bastante tiempo corrigiendo los requisitos funcionales provocando efectos económicos sustanciales en el proyecto.

8.5 Ejemplo de requisitos no funcionales

Vamos a continuar con el caso del sistema que se ha propuesto como ejemplo en donde, después de las preguntas realizadas al cliente se toma la decisión de tomar en cuenta los siguientes atributos de calidad: seguridad, accesibilidad, eficiencia, usabilidad y mantenimiento. Las tablas siguientes muestran los resultados.

Código	RNF-0001
Nombre	Acceso al Sistema
Atributo de calidad	Seguridad
Versión	01.03
Fecha	29/05/2020
Autor	AUT-0004
Fuente	FUE-0007
Descripción	<p>El cliente desea que su aplicación sólo pueda ser accedida por los administradores del negocio, se extrae lo siguiente:</p> <p>El sistema contará con un login de usuario que permita verificar si se trata de un usuario administrador.</p> <p>El sistema deberá permitir el acceso solo a los dos administradores que son dueños del negocio.</p>
Importancia	Vital
Estado	Concluido
Comentario	El presente requisito tuvo origen en la pregunta 31 de la FUE-0007: 31. ¿Desea que su aplicación sólo pueda ser accedida por los administradores?

Tabla 8.1: Requisito no funcional RNF-0001

Código	RNF-0002
Nombre	Accesibilidad de usuarios
Atributo de calidad	Accesibilidad
Versión	01.02
Fecha	28/05/2020
Autor	AUT-0005
Fuente	FUE-0002 FUE-0005 FUE-0007
Descripción	<p>En vista que el cliente manifestó tener deseos de expansión se define la necesidad de contratación de nuevos empleados. Además, se debe tener en cuenta:</p> <p>Accesible a todos los tipos de usuario, incluidas aquellas que presentan habilidades especiales.</p> <p>Consideración a personas con dificultades visuales (daltonismo), dificultades motoras (amputación de algunos dedos).</p>
Importancia	Opcional
Estado	Concluido
Comentario	El presente requisito tuvo origen en las preguntas 35, 41 de la FUE-0007: 35. ¿Tiene pensado extender su negocio, ¿cuánto? y ¿qué actividad tal vez ya está realizando? 41. ¿Usted cree que en algún momento contratará a nuevo personal de manera que tendrá que añadirlo al sistema?

Tabla 8.2: Requisito no funcional RNF-0002

Código	RNF-0003
Nombre	Rapidez en atención y acceso a base de datos
Atributo de calidad	Eficiencia

Versión	01.03
Fecha	29/05/2020
Autor	AUT-0008
Fuente	FUE-0005 FUE-0007
Descripción	<p>El cliente manifiesta tener deseos de expansión, así como tener una mayor afluencia de clientes en la hora de desayuno y almuerzo todos los días, con 20 a 25 clientes por día. Asimismo:</p> <p>El sistema propuesto debe abastecer con la demanda de procesamiento en las horas pico enunciadas por el cliente, además, por ser una bodega pequeña, tiene 25 clientes por día. Cuando se decida incrementar el número de clientes por día, debe evitar el congestionamiento del sistema.</p> <p>La base de datos es un recurso compartido por todas las instancias del sistema. El sistema accederá a la base de datos en reiteradas ocasiones ya sea para actualizar o añadir los registros que se encuentren en ella, por lo tanto, el sistema debe asegurar el descongestionamiento en las consultas.</p> <p>Se debe de especificar condiciones de uso óptimo (Por ejemplo, cantidad de consultas óptimas por minuto).</p>
Importancia	Vital
Estado	Concluido
Comentario	<p>El presente requisito tuvo origen en las preguntas 35, 36 y 38 de la FUE-0007:</p> <p>35. ¿Tiene pensado extender su negocio, ¿cuánto? y ¿qué actividad tal vez ya está realizando?</p> <p>36. ¿Cuántas salidas y/o entradas tiene usted al día o a la semana?</p> <p>38. ¿Qué día y a qué horas es donde hay más afluencia de compradores?</p>

Tabla 8.3: Requisito no funcional RNF-0003

CAPÍTULO 9

Gestión del Riesgo en los Requisitos

9.1 Introducción

Hoy en día, la ingeniería de requisitos es un tema de investigación que trata de resolver una instancia importante que concierne al proceso, es decir, la concepción de lo que se desea producir. Jackson sostiene que la ingeniería de requisitos se encuentra ubicada entre lo informal y lo formal de la construcción del software [52]. Para hacer lo que el cliente necesita, en el tiempo y costo asignado, es necesario diseñar un proceso donde se incluya, en las etapas tempranas de la construcción del software, la gestión de los riesgos asociados a los requisitos, de manera que se contribuya en el mejoramiento de la gestión de un proyecto de software.

Por otro lado, es ampliamente conocido que la práctica de la Ingeniería de Requisitos falla por múltiples razones como los son:

1. Los clientes no tienen claro sobre la complejidad de la construcción de los productos de software, por lo tanto, desconocen la funcionalidad del producto y sobre sus atributos no funcionales.
2. Las necesidades del cliente son dinámicas, por lo que se espera que cambien continuamente dando como consecuencia que los clientes terminen por no comprender bien el producto.
3. Es común entender que los analistas desconocen en su totalidad la forma como funciona la organización.
4. Los clientes siempre desean que el producto sea entregado lo más pronto posible lo que hace que la presencia del interesado en el proyecto sea pobre.
5. Se piensa que el traer los conocimientos de proyectos anteriores entonces la construcción del producto sea más rápido. No se lleva a cabo un análisis de los procesos previos.
6. En los proyectos de software es común que el recojo de información lo realicen personas

sin experiencia. El efecto es grave porque se introducen serios errores de concepción de los requisitos.

7. Normalmente no existe una buena interacción entre el cliente, los analistas y desarrolladores. Esta parca relación no permite conceptualizar una buena visión de las necesidades del cliente.
8. Tampoco se piensa en los usuarios finales. Estos son entrevistados al final por lo que existe una alta probabilidad de introducir serias modificaciones en los requisitos.
9. Los atributos de calidad no son tomados en cuenta en el momento adecuado; estos recién son tomados cuando el producto de software ingresa a la fase de producción.
10. Los requisitos no funcionales son soslayados y quedan relegados para el final. Existe un gran desconocimiento entre la relación que guarda la calidad y los requisitos no funcionales.

Normalmente los riesgos son considerados como una amenaza a la concepción y definición de los requisitos por lo que estos deben ser controlados. Los riesgos en los requisitos deben ser transformados en oportunidades para así no generar errores al momento del diseño del producto. Estos no deben de evitarse, deben ser tomados en cuenta para elaborar requisitos de calidad.

Por ejemplo, en la construcción de un producto de software se detecta que las computadoras donde se va a instalar el producto para ejecutarlo son del tipo Pentium IV; este hecho no debe ser tomado en cuenta para modificar el código sino para dejar áreas en la codificación ya que, en el futuro, el cliente puede adquirir nuevas computadoras que contengan sistemas operativos modernos.

9.2 Definición

Antes de entrar a la definición de riesgo, es necesario conocer conceptos que guardan relación con el mismo. Así podemos mencionar que:

1. Activo: Es cualquier recurso de hardware, software, infraestructura, de personal, de información, administrativo, de comunicación, de documentos entre otros. Ejemplo de activo son: bases de datos, sistema operativo, protocolo de comunicación, aplicaciones, DNI, RUC, plantillas de trabajo, documentos de educación, documento de ilación, documento de especificación, diagrama de clases, diagrama de procesos, entre otros.
2. Vulnerabilidad: Se entiende como la debilidad que puede ser accionada de dos maneras: accidental o intencional. La vulnerabilidad es la causa de riesgo de los componentes comprometidos en una amenaza y pueden experimentar daño. Por ejemplo, en una interfaz gráfica de usuario que suma valores referidos a monedas, si no se especifica el tipo de moneda entonces puede generar interpretaciones erróneas.
3. Amenaza: La amenaza implica que sea ejecutada una vulnerabilidad específica. Estas deben de existir para que generen un riesgo y pueden ser medidas bajo criterios de

probabilidad. Se debe recordar que las amenazas generan daños que pueden causar destrucción, modificación o visualización de datos. Por ejemplo, una falta en la seguridad del código fuente puede causar que otras personas realicen modificaciones en el mismo, de tal forma que pueden alterar los valores.

4. Impacto: El impacto es el daño que un riesgo ejerce sobre un activo. Por ejemplo, cuando las características de un usuario final no son tomadas en cuenta (falla en la toma de un requisito no funcional) y el producto es construido de manera que a los desarrolladores les parece normal, al momento de la utilización del producto, por este usuario, nos damos cuenta que presenta discapacidad en las extremidades superiores.
5. Suposición: Son confirmaciones reales sin sustento alguno que deben ser analizadas cualitativamente, controladas y documentadas. La suposición y el riesgo presentan definiciones similares, aunque algunos autores indican que son un tipo de riesgo, porque participan con dos conceptos en forma común: la probabilidad y el impacto. Las suposiciones con una baja probabilidad e impacto elevado se convierten en riesgos. Por ejemplo, en el caso del ejemplo proporcionado en el impacto, los desarrolladores supusieron que el usuario final se trataba de una persona no discapacitada.

Así, un riesgo se define como la “*Contingencia o proximidad de un daño*” según la RAE, o como el “*evento o condición incierto que, si se produce, tendrá un efecto positivo o negativo sobre al menos un objetivo, como tiempo, costo, alcance o calidad, es decir, cuando el objetivo de tiempo es cumplir con el cronograma acordado; cuando el objetivo de costo es cumplir con el costo acordado, entre otros*” según INTECO [53]. PMI [54] define el riesgo como el “*evento o condición incierta que, si ocurre, tendrá un efecto positivo o negativo en los objetivos del proyecto*”.

9.3 Riesgos en el desarrollo de software

Los riesgos en el desarrollo de software parten de la mala forma de concebir los requisitos. En muchas oportunidades estos son expresados de manera inadecuada de tal forma que el entendimiento por otros profesionales es casi nulo. Los riesgos comúnmente se pueden deducir de:

Riesgos relacionados con las personas

- Falta de experiencia del equipo que desarrolla el producto de software en cuestiones de análisis y diseño.
- Falta de organización del equipo de desarrollo, implica la falta de liderazgo.
- Los usuarios finales no intervienen en la elaboración de las interfaces gráficas de usuario por lo que estas son diseñadas por los propios analistas.
- El equipo desarrollador no cuenta con la infraestructura necesaria para realizar su trabajo, esto hace que la incomodidad les permita obviar actividades consideradas

como importantes.

- La falta de capacitación o conocimiento en el uso de herramientas es otro factor que permite la inserción de riesgos.
- Asumir que los clientes tienen o cuentan con recursos de hardware que comúnmente se usan para el desarrollo, generan riesgos que pueden resultar costosos para el proyecto.
- Los usuarios finales cambian o agregan continuamente requisitos proporcionando inestabilidad en la continuidad del trabajo.
- Los usuarios finales quedan insatisfechos producto de que no han participado, como elementos externos, en la construcción del software o, como puede ser, su escasa participación.
- Aquellos usuarios finales que fueron obligados a participar pueden entregar requisitos inconsistentes o aprobaciones de interfaces que no fueron verificadas.
- La falta de aprobación del sistema por parte de los usuarios finales producto de que no han participado en el proyecto y los analistas han supuesto detalles que consideran normales.
- La capacitación a los usuarios finales es ínfima o escasa.
- Los usuarios finales no desean participar en el proyecto por problemas personales o políticos.
- Los clientes se inmiscuyen en las decisiones técnicas del proyecto de desarrollo.
- Los usuarios finales casi siempre no saben explicar sus necesidades.
- Desarrolladores a los que no se les cancela sus honorarios a tiempo.
- Conflictos entre los integrantes del equipo de desarrollo de software.
- Falta de experiencia para representar requisitos y su relación con el análisis y diseño.
- Una mala selección de los integrantes del equipo desarrollador.
- Inadecuada motivación del equipo desarrollador.
- No se lleva a cabo una buena gestión para enfrentar a integrantes que generan problemas al interior del equipo.
- Incorporación de más personas cuando el proyecto tiene un retraso sustancial o retiro de las mismas cuando se tiene un avance significativo.

Riesgos relacionados con el proceso

- Mal entendimiento de la propuesta y una mala formulación de la planificación de la toma de información.
- Omitir un conjunto de actividades relacionadas con la construcción del software, como por ejemplo elaborar catálogos de requisitos simplificados.

- Los requisitos no son obtenidos de manera clara y absoluta debido a que los usuarios finales no participaron del proyecto.
- Los requisitos no son definidos claramente debido a la falta de liderazgo.
- Un mal dimensionamiento del equipo de desarrollo de software.
- Las planificaciones se hacen de manera optimista pensando que las actividades van a ocurrir tal como se planifican.
- No se lleva a cabo una buena gestión de riesgos o los mismos no son tomados en cuenta.
- Falla en la entrega de subproductos por parte de los proveedores.
- Inadecuada planificación de las actividades del proyecto de desarrollo.
- En muchas oportunidades, debido a la presión ejercida por el cliente, se deja de lado la planificación.
- Desperdicio del tiempo antes de comenzar las actividades reales para el desarrollo de software.
- Diseños inadecuados son tomados como reales sin validarlos por medio de alguna técnica.
- No tomar en cuenta las actividades del control de calidad hacen que el producto contenga errores.
- No se definen los controles para el proyecto incrementando el desorden en la etapa de construcción.

Riesgos relacionados con el producto

- Idealizar los resultados del software implica un alto optimismo en la construcción del producto.
- Desconocimiento en el uso de técnicas, métodos, metodologías, modelos y metamodelos en la construcción de software.
- Falta de involucramiento, en un 100 %, de todos los usuarios del sistema en la construcción del producto de software.
- Los clientes siempre desean que todas sus necesidades se encuentren representadas en el producto.
- La mala calidad del software permite que el producto no genere la confianza del caso en los clientes.
- Dejar de lado el empleo de estándares para la construcción del producto.
- Añadir requisitos que no fueron solicitados por el cliente y supuestos por los desarrolladores.
- Añadir o modificar continuamente los requisitos generan inestabilidad en el control de versiones y en el producto.

- Los desarrolladores añaden características que piensan que el cliente o usuario final las aceptaría.
- El cliente concede retrasos o ampliaciones de tiempo y el desarrollador incrementa requisitos al sistema.

Riesgos relacionados con la tecnología

- El cambio de plataforma donde corre el producto final.
- Confiar demasiado en que una nueva tecnología, metodología, lenguaje o plataforma, puedan resolver los grandes problemas de manera inmediata.
- Obtener el máximo rendimiento de una nueva herramienta o método lleva un tiempo de aprendizaje para usarlo (curva de aprendizaje) y usarlo de manera correcta.
- La falta de automatización del control del código fuente permite una desorganización en el interior de la codificación del producto.
- La falta de disponibilidad de herramientas permite que los integrantes del equipo de desarrollo de software incrementen sus tiempos de construcción.
- Una pésima elección de herramientas de trabajo permite que el equipo de desarrollo solo obtenga resultados parciales teniendo que completar la información en forma manual o con otras herramientas.

9.4 Descripción del proceso

El proceso de gestión de riesgos, cuando se trata de requisitos, se hace en cada una de las etapas que lo conforman. Se identifican, analizan, evalúan y se tratan todos los riesgos asociados con cada una de las etapas de la elicitation. Se deben de revisar cada uno de ellos para consolidar su retroalimentación sin dejar de lado las consultas y el control de versiones de la documentación.

Después de múltiples iteraciones, se controlan los riesgos para finalmente insertarlos en cada una las fases de elicitation y lograr un catálogo de requisitos que responda, de manera adecuada, a la etapa de codificación del producto. La figura 9.1 muestra el proceso.

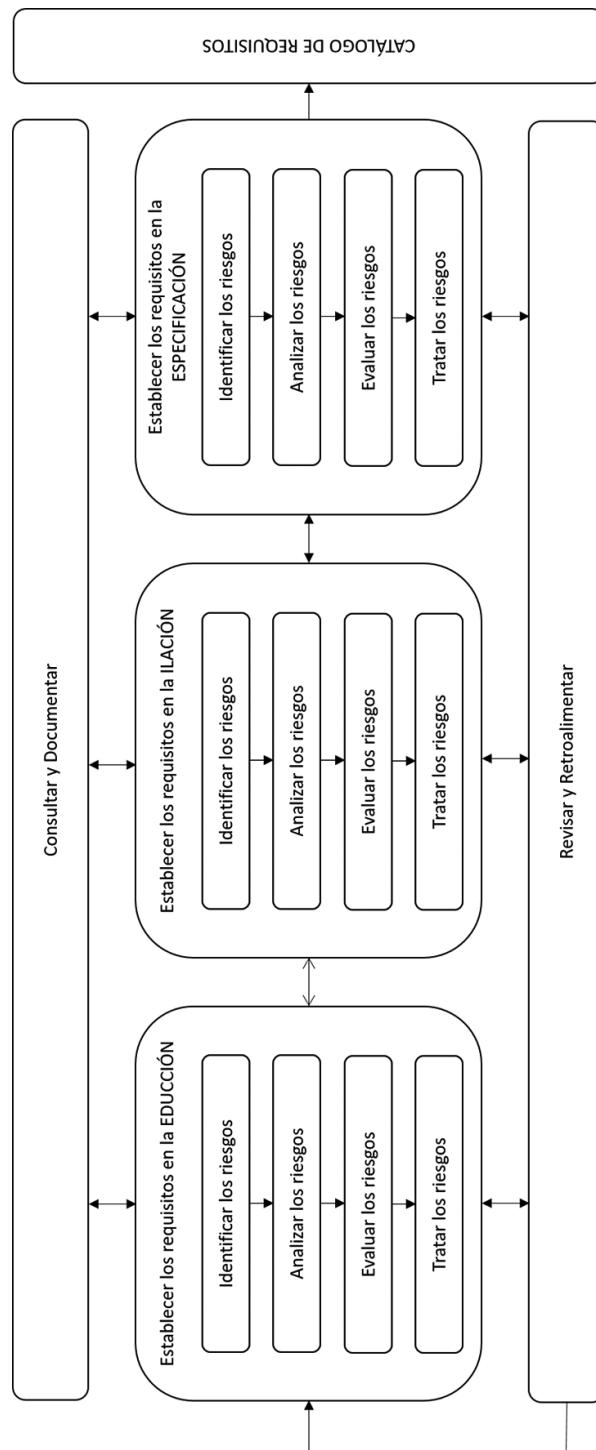


Figura 9.1: Proceso de gestión de riesgos en los requisitos

9.5 Diseño de la plantilla de trabajo

La plantilla donde se plasman los riesgos no presenta las características de aquellas plantillas asociadas con los proyectos. Estas se encuentran adecuadas a la percepción de los requisitos. La figura 9.2 muestra la misma. La primera columna se encuentra relacionada con el código del requisito; es decir que se debe colocar los códigos de cada tabla en cada una de sus etapas como por ejemplo EDU-0001, EDU-0002 cuando se trata de educación.ILA-0001,ILA-0002 cuando se refiere a ilación y ESP-0001, ESP-0002 cuando se trata de la especificación de requisitos.

En la columna denominada como "*Riesgo identificado*" se debe de escribir el riesgo asociado con el requisito. En la columna "*Probabilidad (cuantitativa)*" se debe agregar la probabilidad de que el riesgo ocurra, pero en términos cualitativos como, por ejemplo: muy baja(5 %), baja(10 %), media-baja (25 %), media (50 %), media-alta (75 %), alta (90 %) y muy alta (95 %).

En la columna "*Probabilidad*" se escribe el porcentaje de la probabilidad de ocurrencia, esta se encuentra dentro del rango del 0 % al 100 %. En la columna "*Impacto*" se detalla, en días, el impacto que se tendría en caso de ocurrir un riesgo. En la columna "*Índice P-I*", los valores se obtienen multiplicando la probabilidad por el impacto; el dato se emplea para estimar un colchón del proyecto con los 10 principales riesgos.

En la columna "*Costo por error*" se debe de insertar el valor del costo por error cometido y en la columna "*Exposición al Riesgo*" se calcula automáticamente como resultado del producto del índice P-I por el costo por error cometido. En la columna "*Estado*" se coloca una de tres opciones: retrasado, en proceso o pendiente. En la columna "*Plan de Mitigación*" se coloca el plan de mitigación asociado con el riesgo del requisito.

Finalmente, en la columna "*Código Artefactos*" se agregan los códigos de aquellos artefactos asociados al requisito; esto puede ser una interfaz gráfica de usuario, diagrama de clases, diagrama de secuencia, diagrama de objetos entre otros. Recordar que el estado también puede estar asociado con colores como el amarillo para el proceso, el verde cuando se encuentra resuelto y el rojo cuando se encuentra retrasado.

Plantilla de Gestión de Riesgos

Figura 9.2: Plantilla de gestión de riesgos para requisitos

9.6 Ejemplo de gestión del riesgo

El ejemplo corresponde al problema que se viene tratando a lo largo del texto y muestra solo a un conjunto reducido de requisitos tratados en el catálogo de requisitos (tabla 9.1). En la primera columna se designa el código de la tabla correspondiente (que puede ser la tabla de educación, ilación o especificación), en la tercera columna se muestra el riesgo asociado a la tabla correspondiente; la cuarta columna se encuentra asociada con la probabilidad cualitativa que se otorga al riesgo.

La quinta columna muestra la probabilidad expresada en forma cuantitativa y en la sexta el impacto expresado en días si el riesgo ocurriera. La séptima columna muestra el índice P-I que es calculada automáticamente por la multiplicación de la probabilidad por el impacto, esta información es empleada para estimar un colchón del proyecto con los diez principales riesgos deducidos de los requisitos.

La octava columna muestra el costo, en dólares, por error detectado; la novena columna se hace la exposición al riesgo que es calculada como el producto del índice P-I por el costo por error cometido. Finalmente, la décima columna muestra el estado del requisito, el mismo que puede tener tres características: en proceso, resuelto o retrasado. Dependiendo de ello se debe prestar atención en mayor o menor cuantía al requisito. Las dos últimas columnas muestran el plan de mitigación del requisito y el código del artefacto asociado.

Código del Requisito	Nombre del requisito asociado	Riesgo identificado	Probabil. (calitativa)	Prob.	Imp.	Índice P-I	Costo por error	Exposición al riesgo	Estado	Plan mitigación	Código artefactos
General	General	Corte de energía (batería de los dispositivos)	Baja	10%	0.50	0.050	\$8.93	\$0.45	En proceso	Contar con un mensaje de alerta por batería insuficiente, que le avise la probabilidad de que se apague el celular.	Ninguno
General	General	Corte de internet	Baja	10%	0.08	0.0008	\$1.49	\$0.01	En proceso	Indicar al usuario con una pantalla en la aplicación que diga, "pérdida de conexión a Internet"	Ninguno
General	General	El usuario ingresa caracteres inválidos en los distintos campos de un formulario "registrar" o "modificar" de acuerdo a las convenciones establecidas en cada ilación	Media-baja	25%	2.00	0.500	\$35.72	\$17.86	En proceso	El sistema mostrará mensajes emergentes de cada input del formulario donde se ingresaron datos que no cumplieran las convenciones establecidas en cada ilación.	Ninguno
General	General	Cambio de versión de Android, perdiendo o cambiando el soporte a funciones o tecnologías utilizadas en el proyecto	Muy baja	5%	4.00	0.200	\$71.44	\$14.29	En proceso	Realizar una nueva versión del aplicativo donde solo se actualicen los módulos afectados.	Ninguno
ESP-0001	Menú principal	Error al intentar registrar usuarios mediante un formulario de registro de datos de usuario con campos obligatorios vacíos.	Baja	10%	0.50	0.050	\$8.93	\$0.45	En proceso	El equipo de desarrollo da soporte a mantenimiento por 2 años o hasta que se decida cambiar la aplicación	Ninguno
ESP-0002	Vista de Gestión de usuarios	Interfaz poco amigable con el usuario	Media	50%	2.00	1.000	\$35.72	\$35.72	En proceso	Presentación de prototipo del sistema para ser evaluado por el usuario	Ninguno

ESP-0003	Formulario de registro de usuario	Ingresos inválidos en los campos del formulario	Media	50%	0,50	0,250	\$8,93	\$2,23	En proceso	El sistema envía un mensaje de error indicando un ingreso inválido.	Ninguno
ESP-0004	Registrar un usuario en la base de datos	La base de datos se encuentra llena y no permite el registro de nuevos usuarios.	Media-baja	25%	1,00	0,250	\$17,86	\$4,47	En proceso	Contratación de un servicio de ampliación de capacidad de la base de datos.	Ninguno
ESP-0005	Listar usuarios registrados en la base de datos	No se recuperan los registros de la base de datos.	Media-baja	25%	1,00	0,250	\$17,86	\$4,47	En proceso	El equipo de desarrollo acude a revisar el estado del sistema.	Ninguno
ESP-0006	Formulario de inicio de sesión	El usuario no recuerda sus datos de inicio de sesión.	Media	50%	1,00	0,500	\$17,86	\$8,93	En proceso	El equipo de desarrollo acude a recuperar los datos del usuario.	Ninguno
ESP-0007	Verificación de existencia del usuario	El sistema hace la consulta en la Tabla "Usuario" de la base de datos del sistema y este no encuentra coincidencias.	Media-baja	25%	0,50	0,125	\$8,93	\$1,12	En proceso	El sistema muestra un Alertdialog con el mensaje "El usuario no existe"	Ninguno
ESP-0008	Listar tipos de usuarios	No se recuperan los registros de la base de datos.	Baja	10%	1,00	0,100	\$17,86	\$1,79	En proceso	El equipo de desarrollo acude a revisar el estado del sistema.	Ninguno
ESP-0009	Formulario de modificación de datos de usuario	Cambios en la apariencia del formulario debido a incompatibilidades con la versión de Android.	Media	50%	2,00	1,000	\$35,72	\$35,72	En proceso	Elaboración de manuales de usuario con las especificaciones necesarias para uso del sistema.	Ninguno
ESP-0010	Modificación en BD de un usuario	No se modifican los datos de la base de datos por perdida de conexión	Media	50%	1,00	0,500	\$17,86	\$8,93	En proceso	Mostrar un Toast con el mensaje "Error en actualización de datos de usuario"	Ninguno

Tabla 9.1: Gestión de riesgos para el ejemplo - etapa de especificación

CAPÍTULO 10

Requisitos y Pruebas de Software

10.1 Introducción

En muchas oportunidades se obliga a empezar a determinar el plan de pruebas de software en etapas tempranas de su construcción; para hacer esto imaginamos lo que se puede probar, producto de una visión que se tiene de las interfaces gráficas de usuario. Pocas veces empleamos el catálogo de requisitos para elaborar este plan, producto del desconocimiento o de la falta de experiencia de los analistas que llevan a cabo las pruebas de software.

Es común, en las empresas desarrolladoras de software, llevar a cabo las pruebas de software después de que haya culminado la etapa de codificación del producto y proceden a revisar tanto el código, así como el problema de la funcionalidad del mismo. Al hacer esto, no conceptualizan el tipo de error encontrado y como la modificación en la funcionalidad produce un efecto colateral al producto.

Puesto que el catálogo de requisitos contiene la educación, ilación y especificación, es aconsejable que las pruebas de software se deduzcan de las tablas de especificación, aunque se pueden elaborar grandes bosquejos desde la etapa de educación. El diseñar las pruebas en etapas tempranas de la construcción permite un ahorro sustancial de tiempo y costos, aunque es importante que los diseñadores cuenten con la experiencia del caso.

Fundamentalmente debido a la proliferación de lenguajes de programación, sistemas operativos y elementos de hardware, las pruebas de software se han tornado más complicadas. Aunque el avance tecnológico implica una sofisticación del software, las pruebas de software también han adquirido estas características porque muchos de sus componentes ya han sido probados en entornos específicos.

Se debe de observar que la prueba de software no es nada más que un proceso o un conjunto de procesos que se encuentran definidos para entender que lo que fue codificado es correcto. Las pruebas de software son una tarea técnica que guarda relación con el aspecto económico

y psicológico; y la prueba de una aplicación compleja acarrearía demasiado tiempo porque requiere demasiados recursos humanos para ser económicamente factible.

El probador de software debe tener actitudes adecuadas que le permita tener una visión correcta de lo que tiene que probar. En algunos casos, la actitud del evaluador puede ser más importante que el proceso en sí, ya que tiene que vislumbrar situaciones que permitan ahorrar dinero y proporcionar seguridad en la funcionalidad del producto. Esta experiencia le permita conocer lo que “tiene que probar”.

Una definición apropiada se puede obtener de G. Myers et al. [55] quienes indican que: “*La prueba de software es el proceso de ejecutar un programa con la intención de encontrar errores*”. Si el objetivo es demostrar que la aplicación no contiene errores entonces se seleccionan datos de pruebas que tengan una alta probabilidad de causar que falle la aplicación para lograr correcciones consistentes y guiados por los siguientes principios:

1. Todo caso de prueba debe proporcionar un resultado.
2. Toda aplicación debe ser probada por terceras personas.
3. Toda empresa desarrolladora de software no debe probar sus propias aplicaciones.
4. Todo proceso de prueba debe incluir un análisis de los resultados.
5. Todo caso de prueba debe incluir condiciones para entradas válidas o inválidas.
6. Probar un aplicativo implica observar que no haga lo que debe de hacer, así como observar que el código haga lo que no debe de hacer.
7. Todo caso de prueba desecharable debe ser evitado.
8. Toda prueba de software no debe ser planificada con supuestos tácitos.
9. Si en un área se han encontrado errores entonces es altamente probable seguir encontrando más errores.
10. Toda prueba de software es una tarea creativa.

Los tres métodos más empleados son las inspecciones de código, los recorridos o caminatas y las pruebas de usabilidad. Existen textos especializados en estos temas donde hacen una larga y profunda explicación sobre pruebas de software, sus metodologías y la forma de aplicación. El objetivo que persigue el presente texto es el de saber qué pruebas se aplican.

Existe una gran variedad de pruebas de software y su clasificación se muestra a continuación:

1. Prueba de caja negra: Técnica en la que se verifica la funcionalidad del producto sin tomar en cuenta la estructura del código y sus detalles de implementación.
2. Prueba de caja blanca: Técnica que se centra en la estructura del código. Los probadores de software introducen valores iniciales para probar los flujos de ejecución.
3. Prueba de complejidad ciclomática: Técnica que entrega indicadores que demuestran la complejidad lógica de un programa.
4. Prueba unitaria: Técnica que prueba unidades de código, o sea funciones, procedimientos

o clases de manera separada y que permite comprobar la eficiencia de los mismos en forma independiente.

5. Prueba de aceptación: Técnica que se emplea antes de liberar el producto de software con la finalidad de determinar si resuelven las necesidades de los usuarios.
6. Prueba de integración: Técnica empleada después de haber llevado a cabo las pruebas unitarias con la finalidad de conocer de que juntas funcionan eficientemente.
7. Prueba de regresión: Técnica que evalúa el correcto funcionamiento del software frente a evoluciones o cambios funcionales con la finalidad de descubrir errores o carencias de funcionalidad.
8. Prueba de carga: Técnica que permite observar el comportamiento de una aplicación bajo una petición considerable de datos.
9. Prueba de estrés: Técnica que permite romper la aplicación con la finalidad de conocer la solidez de la misma.
10. Prueba de escalabilidad: Técnica que permite conocer la fortaleza de los requisitos no funcionales para determinar el grado de escalabilidad de una aplicación.
11. Prueba de portabilidad: Técnica que permite conocer el grado de dificultad de una aplicación al ser transferido de un sistema operativo a otro.
12. Prueba de seguridad: Técnica orientada a los requisitos no funcionales que permite constatar su desempeño en función de los requisitos funcionales.
13. Prueba de interoperabilidad: Técnica que permite conocer las deficiencias del traslado de información entre dos o más aplicaciones.
14. Prueba de componente: Técnica que permite validar si los componentes integrados a la aplicación funcionan correctamente.
15. Prueba de compatibilidad: Técnica que permite conocer las deficiencias de las aplicaciones cuando estas se mueven en diferentes entornos.
16. Prueba de usabilidad: Técnica que permite llevar a cabo las pruebas de software directamente con los usuarios finales del sistema.
17. Prueba de rendimiento: Técnica que permite determinar la rapidez de respuesta de un aplicativo en condiciones particulares.
18. Prueba de configuración: Técnica que permite validar y verificar que los sistemas funcionan correctamente en las estaciones de trabajo seleccionadas.
19. Prueba de conversión: Técnica que permite validar la conversión de los datos en sistemas antiguos y su real valor en los sistemas nuevos.
20. Prueba de instalación: Técnica que permite validar que los aplicativos se instalen correctamente en los dispositivos seleccionados.

Existen una serie de metodologías que se pueden emplear, por ejemplo para la prueba de complejidad ciclomática se puede utilizar: clases o particiones de equivalencia, análisis del valor límite, grafos de causa y efecto y error de adivinación. Asimismo, se pueden emplear

las siguientes metodologías para las pruebas de caja blanca: cobertura de declaraciones, cobertura de decisiones, cobertura de condiciones, cobertura de decisión-condición y coberturas de condiciones múltiples. La figura 10.1 muestra una plantilla para pruebas de software.

Organización		
Autor		
Versión		
Etapa		
Fecha		
Código del requisito	Descripción de la prueba	Tipo

Figura 10.1: Plantilla de pruebas de software para requisitos

10.2 Pruebas de software y el catálogo de requisitos

Existen dos tipos de diseño del plan de pruebas, el primero se encuentra orientado al catálogo de requisitos y relacionadas con las pruebas iniciales y el segundo relacionadas con las pruebas específicas y finales.

Las pruebas iniciales permiten tener una idea general de las pruebas que se pueden llevar a cabo para finalmente definir las pruebas definitivas. Los desarrolladores toman esto como una pérdida de tiempo; lo que se puede indicar es que cuando no se tiene experiencia se puede comenzar por lograr un listado completo de las pruebas a realizar a partir del catálogo para finalmente determinar donde se llevarán a cabo las pruebas finales.

Para el caso del catálogo de requisitos, se puede emplear un diseño básico tomando como base la especificación de requisitos. En vista de que no existe un código definido, por la lectura del mismo se puede establecer el tipo de prueba de manera genérica para que cuando se genere el código se haga una revisión exhaustiva. Estas pruebas son consideradas iniciales porque no tienen el análisis profundo adecuado. La figura 10.1 muestra un primer diseño.

En “Organización” se coloca el código de la organización para quien se está preparando la aplicación (ORG-DDDD). En el campo “Autor” se escribe el código de la persona que llevará a cabo la prueba de software (AUT-DDDD). Asimismo, se agrega la versión (VDDD). En “Etapa” se escribe con cuál de las etapas del catálogo de requisitos se va a trabajar (educción, ilación, especificación) para finalmente escribir la fecha cuando se planifican las pruebas de software.

En el “Código del requisito” se escribe el código de la tabla desde donde se encuentra escrito el requisito, por ejemplo, si se decide trabajar con la tabla de especificación se debe de escribir ESP-DDDD, si es ilación ILA-DDDD o si es educación EDU-DDDD; en “Descripción de la prueba” se detalla la prueba de software que se llevará a cabo.

10.3 Ejemplo de la definición de pruebas de software

A continuación, se presenta las pruebas de software en la etapa de especificación del ejemplo que se presenta a lo largo del texto. Aunque no se pretende hacer un tratado sobre pruebas de software, se presentan a continuación un listado de ellas según la especificación de requisitos (tabla 10.2). En la tabla 10.1 se muestra una codificación de las pruebas de software.

Nro.	Tipo de prueba	Nombre
1	P001	Prueba de caja negra
2	P002	Prueba de caja blanca
3	P003	Prueba de complejidad ciclomática
4	P004	Prueba unitaria
5	P005	Prueba de aceptación
6	P006	Prueba de integración
7	P007	Prueba de regresión
8	P008	Prueba de carga
9	P009	Prueba de estrés
10	P010	Prueba de escalabilidad
11	P011	Prueba de portabilidad
12	P012	Prueba de seguridad
13	P013	Prueba de interoperabilidad
14	P014	Prueba de componente
15	P015	Prueba de compatibilidad
16	P016	Prueba de usabilidad
17	P017	Prueba de rendimiento
18	P018	Prueba de configuración
19	P019	Prueba de conversión
20	P020	Prueba de instalación

Tabla 10.1: Codificación de las pruebas de software

Código del Requisito	Nombre del requisito asociado	Descripción de la prueba	Tipo	Descripción
ESP-0001	Menú principal	Tiempo de carga de interfaz	P008	Prueba de carga
		Gráfico de causa-efecto para determinar la navegabilidad desde el menú principal	P001	Prueba de caja negra
		Probar que todos los componentes de la interfaz se visualicen de forma correcta en distintos dispositivos	P015	Prueba de compatibilidad
		Presentar al cliente la interfaz de usuario	P005	Prueba de aceptación
ESP-0002	Vista de gestión de usuarios	Gráfico de causa-efecto para determinar la navegabilidad desde la Vista de gestión de usuarios	P001	Prueba de caja negra
		Probar que todos los componentes de la interfaz se visualicen de forma correcta en distintos dispositivos	P015	Prueba de compatibilidad
		Presentar al cliente la interfaz de usuario	P005	Prueba de aceptación
ESP-0003	Formulario de registro de usuario	Prueba de camino básico para evaluar esta funcionalidad	P002	Prueba de caja blanca
		Partición equivalente sobre campos del formulario	P001	Prueba de caja negra
		Probar que todos los componentes de la interfaz se visualicen de forma correcta en distintos dispositivos	P015	Prueba de compatibilidad
		Validación de los datos ingresados en los campos del formulario	P004	Prueba unitaria
		Presentar al cliente la interfaz de usuario	P005	Prueba de aceptación
ESP-0004	Registrar un usuario en la base de datos	Cálculo del tiempo de registro de un usuario en la base de datos	P001	Prueba de caja negra
ESP-0005	Listar usuarios registrados en la base de datos	Prueba de camino básico para evaluar esta funcionalidad	P002	Prueba de caja blanca
		Cálculo del tiempo de carga del listado	P008	Prueba de carga
ESP-0006	Formulario de inicio de sesión	Prueba de camino básico para evaluar esta funcionalidad	P002	Prueba de caja blanca
		Probar que todos los componentes de la interfaz se visualicen de forma correcta en distintos dispositivos	P015	Prueba de compatibilidad
		Partición equivalente sobre campos del formulario	P001	Prueba de caja negra
		Presentar al cliente la interfaz de usuario	P005	Prueba de aceptación
ESP-0007	Verificación de existencia del usuario	Prueba de camino básico para evaluar esta funcionalidad	P002	Prueba de caja blanca
		Cálculo del tiempo de consulta	P008	Prueba de carga
ESP-0008	Listar tipos de usuarios	Cálculo del tiempo de consulta y visualización de los tipos de usuario	P008	Prueba de carga

ESP-0009	Formulario de modificación de datos de usuario	Prueba de camino básico para evaluar esta funcionalidad	P002	Prueba de caja blanca
		Partición equivalente sobre campos del formulario	P001	Prueba de caja negra
		Probar que todos los componentes de la interfaz se visualicen de forma correcta en distintos dispositivos	P015	Prueba de compatibilidad
		Presentar al cliente la interfaz de usuario	P005	Prueba de aceptación

Tabla 10.2: Especificación de requisitos y las pruebas de software

CAPÍTULO 11

Métricas y Requisitos

11.1 Introducción

Las métricas son muy poco empleadas por los desarrolladores de software o por líderes de los proyectos de software. Normalmente, toman otras medidas que les permitan obtener indicadores para llevar a cabo el proyecto. Por otro lado, es común obtener indicadores del proyecto y del producto, pero no de los procesos ya que estos se encuentran directamente dirigidos a los quehaceres y son considerados como “*poco aportante*” para el proyecto.

En la investigación de S. Zapata et al. [56] se lleva a cabo una propuesta para medir la calidad del catálogo de requisitos a partir de la completitud y corrección, mientras que N. Nakatani et al. [57] lo hace a partir de la volatilidad de los requisitos. Investigaciones como las de J. Palmer et al. [58] proponen solo métricas para identificar requisitos en riesgo. Finalmente, estudios de A. Dutoit et al. [59], A. Abarragans et al. [60] y A. Zin et al. [61] proponen métricas para los procesos de comunicación y negociación con los expertos del negocio.

Las investigaciones valoran la importancia de tales medidas ya que afectan a la calidad del proceso y del producto. Por ejemplo, es importante conocer el tiempo que se demora en entender el dominio del negocio y en plasmar un requisito, así como el tiempo que se demora en detectar y corregir las inconsistencias y ambigüedades de los mismos; esto puedecurrir en afectación de los tiempos y costos de construcción de un producto de software.

Tal como se menciona en S. Group [62], una de las causas principales por la cual fracasan los proyectos de desarrollo de software son los requisitos, por lo que esta es una etapa importante y crucial ya que se decide lo que se va a desarrollar. En esta etapa se identifica el contexto y dominio de la aplicación, así como los objetivos generales y específicos del proyecto y las necesidades del cliente.

Para llegar al nivel de evaluación, es preciso contar con datos relevantes, precisos y actualizados sobre diferentes áreas, que faciliten una perspectiva global de la solución. Así,

las métricas de calidad de software pueden aplicarse a diferentes contextos, como lo son: el producto, el proceso y las personas.

El presente texto proporciona un punto de vista para tratar la forma de cómo obtener métricas en la etapa de requisitos (por cierto, estas pueden ser halladas en la etapa de educación, ilación o especificación). Los cálculos de estas medidas permiten ahorrar tiempo de construcción en proyectos futuros permitiendo un ahorro en los costos del mismo.

11.2 Definición

Según la RAE [3] métrica se define como: “(adj) Perteneciente o relativo al metro (unidad de longitud)” y “(f) Arte que trata de la medida o estructura de los versos, de sus clases y de las distintas combinaciones que con ellos pueden formarse.”. En relación a la segunda definición de la RAE se puede concluir que una métrica es cualquier medida orientada a conocer o estimar el tamaño u otra característica de un software o un sistema de información con la finalidad de realizar comparativas para la planificación de proyectos de desarrollo de software.

11.3 Proceso de elaboración de métricas

El proceso comienza con la toma de información que se permite capturar desde los requisitos, entendidos estos, se procede a la formulación de las respectivas métricas. Posteriormente se analiza el requisito y su correspondiente métrica, si estos son correctos, pasan al repositorio caso contrario se valida el requisito y la métrica y se hacen las correcciones del caso incluido el control de versiones y la información para finalmente volver a formular el requisito. La figura 11.1 muestra el proceso.

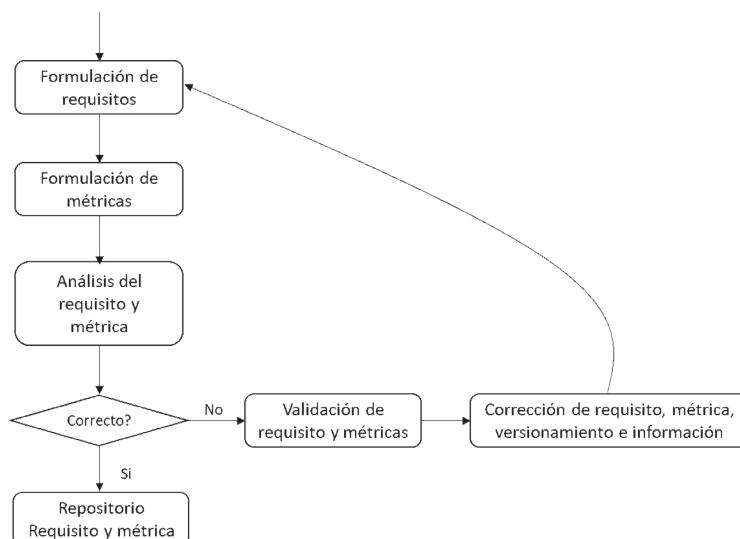


Figura 11.1: Proceso de definición de métricas

11.4 Métricas sugeridas

A continuación, se muestra y explica las métricas sugeridas que se encuentran orientadas a la toma de requisitos:

1. Para determinar la especificidad de los requisitos: Esta métrica se encuentra basada en la coherencia de ideas de los revisores para cada uno de los requisitos. El entendimiento y la percepción juegan un rol importante por lo que se puede determinar calcular el tiempo que se demora en la concepción del requisito visto por diferentes personas. Aunque esto es muy poco frecuente ya que en los proyectos no se destinan dos revisores para un requisito, puede ayudar en el momento en que los revisores interceptan el conjunto de requisitos dentro de sus tareas habituales. Una métrica puede ser:

$$Qr = \frac{A}{B}$$

Donde A es el número de requisitos en la que todos los revisores tuvieron interpretaciones idénticas, B el número total de requisitos. El resultado del valor de Qr debe estar lo más cercano a 1 que implica que el requisito no presenta ambigüedades.

2. Para corrección de requisitos: Una métrica comúnmente sugerida es la proporción de requisitos correctos encontrados en la formación del catálogo de requisitos. Se debe entender que esta proporción tiende a 1 a medida que se lleva a cabo la corrección en función del control de versiones. Cuando el indicador no entregue el valor de 1 significa que los requisitos son correctos. La métrica puede ser:

$$Qc = \frac{A}{(A + B)}$$

Donde A es la cantidad de requisitos correctos y B es la cantidad de requisitos aún no verificados.

3. Para consistencia de la especificación: Esta métrica se encuentra orientada a conocer que tan correcta es una especificación del requisito o que tan concisa es ella. Se sugiere el siguiente indicador:

$$Qs = \frac{1}{(T + 1)}$$

Donde T es la cantidad de versiones empleadas para realizar la especificación de un requisito. Los investigadores sugieren que un valor de respuesta adecuado es 0.1.

4. Para determinar la compresión de los requisitos: Esta métrica se encuentra orientada a conocer si el significado del requisito es comprendido por los usuarios o interesados. La métrica sugerida es:

$$Qu = \frac{A}{B}$$

Donde A es la cantidad de requisitos comprendidos por todos los interesados y B es la cantidad total de requisitos obtenidos incluidos los requisitos no funcionales. Si el resultado es 1 implica el entendimiento de todos los requisitos por parte de los interesados.

5. Otras métricas: Se pueden obtener otras métricas, como, por ejemplo, proporción de requisitos correctamente educacionados, proporción de requisitos correctamente ilados, proporción de requisitos correctamente especificados, tiempo promedio en obtener requisitos educacionados, tiempo promedio en corregir requisitos educacionados, tiempo promedio en implementar el catálogo de requisitos, entre otros.

Para otros analistas puede ser relevante obtener el tiempo promedio que se tarda en lograr una correcta especificación de requisitos o una total construcción del catálogo de requisitos, así como el tiempo que se demora un analista en implementar o validar un requisito.

11.5 Ejemplo de cálculo de métricas

Continuando con el ejemplo propuesto desde el inicio del texto, se muestra el cálculo de métricas para la etapa de educación de requisitos. La tabla 11.1 muestra los resultados. En esta tabla se indica, por cada tipo de actividad, la hora inicial y final de dicha actividad; en la penúltima columna se muestra la cantidad de tiempo consumido y en la última columna se calcula el promedio del tiempo consumido por cada una de estas actividades.

La tabla 11.2 nos indica que se han encontrado un total de 12 requisitos en la etapa de educación y que finalmente se han reducido producto de la fusión que han sufrido tres de ellos quedando al final solo nueve. Por otro lado, se han detectado inicialmente 5 inconsistencias en estos requisitos, así como 4 ambigüedades. De los primeros todos han sido corregidos, así como el de los segundos. Al final sólo quedan nueve requisitos educacionados.

A partir de esta información, podemos calcular una métrica de proporcionalidad con respecto a la cantidad de requisitos educacionados, como, por ejemplo:

$$R = \frac{(A + B)}{T}$$

Donde A representa la suma de la cantidad de requisitos que presentaron inconsistencias y B la cantidad de requisitos que presentaron ambigüedades. T es el número total de requisitos educacionados. Un resultado mayor a la unidad implica la existencia de un gran problema en la toma de información para plasmar los requisitos; y un valor cercano a cero implica lo contrario.

Por otro lado, la última columna de la tabla 11.1 indica que en promedio se tardaron 20 minutos en la entrevista que les permitía corregir las inconsistencias y 20 minutos en la entrevista para corregir las ambigüedades. Por otro lado, gastaron en promedio 69 minutos para educacionar todos los requisitos; así como 7,8 minutos con ocho segundos para detectar todas las inconsistencias.

También se puede interpretar que se tardaron 4 minutos en corregir todas las ambigüedades y 10 minutos en la corrección de inconsistencias; en la corrección de ambigüedades tardaron 4 minutos. Estos indicadores pueden servir para que en el siguiente proyecto de construcción de software podamos contar con otros ingenieros de software para hacer esta tarea.

Claramente se puede notar que los analistas no tienen la suficiente experiencia para llevar a cabo esta actividad, y esto sucede porque no lograron entender el modelo de negocio del cliente o que el cliente no supo entregar la información adecuada como para resolver el problema de la automatización de las tareas. También es probable que existan otros factores, como, por ejemplo, la falta de experiencia de los analistas.

Actividad	Nombre	Hora		Diferencia	Horas	Minutos	Total minutos	Prom.
		Inicio	Fin					
Entrevista general		3:30:00	3:50:00	0:20:00	0	20	20	20
Requisitos		TIEMPO DE ENTREVISTAS PARA CORREGIR INCONSISTENCIAS						
EDU-0001	Gestión de Usuarios	21:00:00	21:20:00	0:20:00	0	20	20	20
EDU-0002	Gestión de Categorías	21:20:00	21:40:00	0:20:00	0	20	20	20
EDU-0003	Gestión de Productos	21:40:00	22:00:00	0:20:00	0	20	20	20
EDU-0004	Gestión de Proveedores	22:00:00	22:20:00	0:20:00	0	20	20	20
EDU-0005	Gestión de Alertas	22:20:00	22:40:00	0:20:00	0	20	20	20
EDU-0006	Gestión de Abastecimiento	22:40:00	23:00:00	0:20:00	0	20	20	20
EDU-0007	Gestión de Salidas	23:00:00	23:20:00	0:20:00	0	20	20	20
EDU-0008	Gestión de Reportes	23:20:00	23:40:00	0:20:00	0	20	20	20
EDU-0009	Gestión de Estadísticas	23:40:00	24:00:00	0:20:00	0	20	20	20
Requisitos		TIEMPO DE ENTREVISTAS PARA CORREGIR AMBIENTES						
EDU-0001	Gestión de Usuarios	21:00:00	21:20:00	0:20:00	0	20	20	20
EDU-0002	Gestión de Categorías	21:20:00	21:40:00	0:20:00	0	20	20	20
EDU-0003	Gestión de Productos	21:40:00	22:00:00	0:20:00	0	20	20	20
EDU-0004	Gestión de Proveedores	22:00:00	22:20:00	0:20:00	0	20	20	20
EDU-0005	Gestión de Alertas	22:20:00	22:40:00	0:20:00	0	20	20	20
EDU-0006	Gestión de Abastecimiento	22:40:00	23:00:00	0:20:00	0	20	20	20
EDU-0007	Gestión de Salidas	23:00:00	23:20:00	0:20:00	0	20	20	20
EDU-0008	Gestión de Reportes	23:20:00	23:40:00	0:20:00	0	20	20	20
EDU-0009	Gestión de Estadísticas	23:40:00	24:00:00	0:20:00	0	20	20	20
Requisitos		TIEMPO PARA LA EDUCACIÓN DE REQUISITOS						

EDU-007	Gestión de Salidas	11:28:00	11:30:00	0:02:00	0	2	2
EDU-006	Gestión de Abastecimiento	12:50:00	12:53:00	0:03:00	0	3	3
EDU-003	Gestión de Productos	11:46:00	11:48:00	0:02:00	0	2	2
EDU-002	Gestión de Categorías	13:00:00	13:10:00	0:10:00	0	10	10

Tabla 11.1: Métricas para la educación de requisitos

Tipo	Inicial	Final	Separados	Fusionados	Corregidos	Total
Educciónados	12	9	0	3	0	9
Inconsistencias	5	0	0	0	5	0
ambigüedades	4	0	0	0	4	0

Tabla 11.2: Cantidad de requisitos tratados en la etapa de educación

CAPÍTULO 12

Inconsistencias y Ambigüedades

12.1 Introducción

S. Group [63] en su investigación lleva a cabo el análisis de los requisitos de software para luego implementar un producto de software que realice esta tarea. Hace una comparativa de los productos de software existentes en el mercado; después de analizar sus ventajas y desventajas concluye que todos estos productos tienen falencias en el sentido de que no introducen todas las funcionalidades de una verdadera Gestión de Requisitos y no toman en cuenta la solución de inconsistencias de los requisitos y otros factores como la trazabilidad.

F. García et al. [64] indican que la Gestión de Requisitos se ha convertido en una herramienta estratégica para la construcción de productos de software ya que su buena gestión proporciona calidad al producto en su ciclo de vida del desarrollo. Concluyen que la Gestión de Requisitos es importante debido, primordialmente, a las inconformidades e inconsistencias surgidas entre los actores principales del producto y mencionan que se deben de tomar bien presente las fases de elicitation, especificación, análisis y validación de requisitos con la finalidad de separar las inconsistencias y ambigüedades.

R. Saavedra et al. [65] Indican que la gran cantidad de problemas que se presentan en la construcción del software se producen por la mala forma de tomar los requerimientos produciendo catálogos de requisitos que contienen inconsistencias y ambigüedades, logrando que estos defectos sean introducidos en fases subsiguientes de la construcción o en su defecto una pérdida de tiempo en la retroalimentación. Asimismo, mencionan que la toma de información mantiene ciertos niveles de complejidad que hacen que se introduzcan una serie de defectos. Finalmente proporcionan un conjunto de estrategias que pueden ser tomadas en cuenta para resolver esta problemática.

F. Levy et al. [66] llevaron a cabo un experimento en 26 estudiantes para lograr determinar el nivel de entendimiento de la toma de información cuando se trata de armar un catálogo de requisitos. Como resultado obtuvieron que los estudiantes empleaban los casos de uso

para lograr una adecuada representación, pero en el análisis de los casos de uso encontraron ambigüedades producto de las inconsistencias de los requerimientos. Concluyen que los estudiantes tienen una especial tendencia a emplear casos de uso y en promedio demoran el mismo tiempo.

J. Valaski et al. [67] llevan a cabo un análisis de todos los trabajos de investigación que han sido presentados en (Workshop of Engineering Requirements) a lo largo de sus años de existencia. Concluyeron que fueron presentados 258 trabajos de investigación y que de ellos el 31 % fueron dedicados al modelamiento de los requerimientos, el 28 % al tema de elicitation y el 16 % a definir métodos o procesos orientados a los requisitos. Solo el 2 % de estas investigaciones estuvieron orientadas a hacer un análisis de los requisitos y al modelo del negocio sin concluir sobre las inconsistencias o ambigüedades introducidas en ellas.

M. Mariduena et al. [68] hacen hincapié en que la calidad de la construcción de software se encuentra relacionado con las necesidades del cliente y la forma de ser satisfechos los usuarios finales por medio de requisitos. Para lograr satisfacer estos criterios de información se debe llevar a cabo un correcto interés en la toma de información. Los requisitos vistos como una verdadera traducción de las necesidades implican el manejo de ciertos factores de transformación de información. Uno de estos factores es la priorización de la información en función de requisitos de software ya que los métodos tradicionales no encuadran este criterio como parte de la construcción del software.

J. Miguetti et al. [69] mencionan que la difusión del software se está acrecentando en las organizaciones; sus ventajas competitivas permiten el ahorro económico, así como el incremento en la velocidad de cómputo logrando resultados a tiempo. Esta disminución de la velocidad es compleja por la calidad de los algoritmos que se deben desarrollar al interior de la construcción y por las amenazas que se reciben cuando de comunicación se trata. Estos factores influyen en la calidad de los requisitos de software ya que se insertan otros factores cuando de telecomunicaciones se trata.

J. Moreno et al. [70] mencionan que los escenarios cobran relevancia cuando se trata de obtener información para elaborar los requisitos; más aún a partir de ellos se pueden obtener patrones que puede hacer que esta actividad tome un tiempo mucho más corto cuando se trata de entender el modelo del negocio. Aseveran que no hay una relación entre escenario y el modelo de negocio por lo que en su trabajo de investigación proponen un conjunto de actividades que permite ir desde un escenario hasta la conceptualización del modelo de negocio. Para hacer esto se deben de encontrar un conjunto de patrones que permite lograr un modelo conceptual adecuado.

J. Rojas et al. [71] mencionan que uno de los problemas más álgidos en la construcción de software es la recolección de la información y el conocimiento de parte de los analistas para lograr entender ello. Su estudio trata de entender la realidad de lo que sucede en Latinoamérica en lo que respecta a la construcción de software y como las organizaciones

desarrolladoras soslayan la etapa de la ingeniería de requisitos. Llevan a cabo un estudio empírico, a través de encuestas y entrevistas a 35 empresas de desarrollo de software cuyos resultados se encuentran orientados a la etapa inicial del análisis de los sistemas y concluyen que estas empresas no prestan la suficiente atención a la ingeniería de requisitos.

P. Huertas [72] presenta la propuesta de un metamodelo de seguimiento para reducir las ambigüedades de los requisitos de software. Este metamodelo proviene de la dificultad de obtener elementos contundentes para concebir el modelo del negocio de un sistema de información. En teoría, la actividad está diseñada como una unidad que define un marco para el análisis de las acciones llevadas a cabo por los analistas. El metamodelo posee puntos de referencia que ayudan a la organización del proceso de educación,licitación y especificación de requisitos de software, que se organiza en torno al concepto de procesos de software. La referencia teórica de este trabajo se centra en la Ingeniería de Requisitos.

12.2 Definiciones

Según la RAE[3], inconsistencia significa: "falta de duración, estabilidad, solidez" y ambigüedad, "Que puede entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión"; lo que implica que la inconsistencia significa que los requisitos no están lo suficiente maduros como para otorgarles la calidad de definitivos y la ambigüedad conlleva a tener definiciones de actividades en diferentes tablas o que dentro de una de ellas existan descripciones paralelas.

12.3 Proceso de corrección

El proceso de corrección implica un análisis de las tablas de educación, ilación y especificación. Cuando en las tablas de educación se encuentran estos defectos entonces se procede a corregirlos versionando las mismas hasta corregir los errores. Los contenidos de las tablas de educación comúnmente se encuentran relacionados con los módulos a construir y pueden refinarse cuantas veces sea posible hasta quedar definidos los límites del sistema.

Las tablas de ilación deben ser analizadas bajo dos ángulos de vista: la primera de manera interna y la segunda buscando su asociación con las tablas de educación. En la forma interna se buscan los defectos y estos son corregidos versionando las correspondientes tablas; en la segunda se estipula que si se encuentra el error interno esto debe ser corroborado con la tabla de educación asociada, si se encuentra un error esta tabla de educación también debe ser corregida y versionada.

Las tablas de especificación siguen el mismo procedimiento. Primero se analizan y corrigen los errores de manera interna versionando estas; segundo se corrigen los errores asociados con las tablas de ilación y si estas presentan correcciones se busca corregir también las tablas de educación. Siempre se emplea el control de versiones que permite guardar el historial

de los cambios propuestos. El proceso de corrección interno se muestra en la figura 12.1.

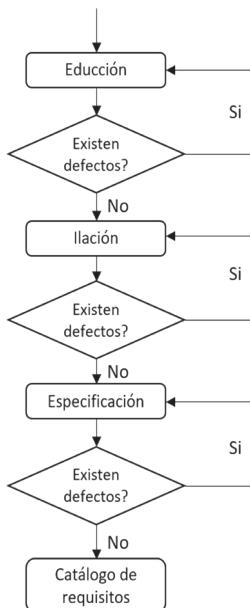


Figura 12.1: Proceso de corrección interno

12.4 Procedimiento para corregir en la educación

El primer momento es crucial cuando se decide elaborar un producto de software para un cliente. En la mayoría de las oportunidades, los analistas desconocen los procesos, procedimientos y actividades de la organización donde se desarrollará, por lo tanto, se tiende primero a conocer el negocio.

El consumo de este conocimiento implica un conjunto de entrevistas con los actores principales del negocio o con quien el cliente asigne para esta tarea. Las formalidades conllevan a conocer la forma como fluye la información dentro de la organización y más aún, dentro del área a automatizar.

Un conjunto de entrevistas es planificado a lo largo del ciclo de vida de la obtención de los requisitos. Para lograrlo se emplean técnicas, metodologías o modelos que permitan obtener esta importante información sin lograr la minuciosidad del caso, pero sin entender los procesos.

Puesto que las necesidades son entregadas en forma bruta entonces se entregan requerimientos que presentan un conjunto de inconsistencias o ambigüedades producto de la mala comprensión de la información o de malas explicaciones por parte de los entrevistados al momento de asumir esta tarea.

El procedimiento que permite obtener la información necesaria para la educación de los requisitos es:

1. Definir e identificar a los actores más importantes dentro de la organización.
2. Definir e identificar a los actores y sus relaciones más importantes dentro del área a automatizar.
3. Planificar las fechas de entrevista con los personajes más importantes en la organización.
4. Definir guiones para las entrevistas asociados con el ámbito del sistema a automatizar.
5. Ejecutar las entrevistas separando hechos relevantes de aquellos que merecen una mejor explicación.
6. Documentar la educación de cada requisito en su correspondiente plantilla.
7. Revisar la educación del requisito definido respetando las opiniones vertidas por los clientes.
8. Detectar inconsistencias o ambigüedades.
9. Corregir inconsistencias o ambigüedades.
10. Versionar la educación de requisitos hasta eliminar las inconsistencias o ambigüedades.

Las técnicas por emplear en esta etapa están asociadas a la complejidad de la toma de información y pueden ser:

- Lluvia de ideas: Un conjunto de personas propone criterios del modelo de negocio y un analista toma lista de ellas.
- Puntos de vista: Los clientes y analistas toman nota de los puntos de vista de cada una de las personas que intervienen.
- Escenarios: La información se toma en función de un conjunto de escenas que conforman el modelo del negocio.
- Episodios: Las escenas contienen episodios y estos son la fuente principal de los analistas para construir el catálogo de requisitos.
- Eventos: En muchas oportunidades los eventos juegan un rol importante para la determinación de requisitos. Estos deben ser analizados y discutidos con el equipo de desarrollo.
- Casos de uso: Ampliamente conocidos y empleados por su flexibilidad y versatilidad. Es una buena forma de documentar los requisitos.
- Etnografía: El tener cuidado con el aspecto social de los usuarios y las personas que componen el equipo de desarrollo es importante ya que se puede convertir en un fracaso en el entendimiento del modelo del negocio.

12.5 Procedimiento para corregir en la ilación

La etapa de ilación de requisitos consiste en que el analista logre descripciones de los requisitos producto del entendimiento de las complejidades del flujo de información y su asociación con los procesos y procedimientos de la organización y del área a automatizar.

Esta etapa se encuentra dedicada específicamente para los analistas quienes deben expresar su entendimiento del modelo del negocio para proponer alternativas de solución que se encuentren de acuerdo con los objetivos de la organización y el objetivo de la automatización.

Este entendimiento lo logra producto del conocimiento adquirido en la etapa de educación de requisitos y de aquellos artefactos que fueron encontrando a lo largo de la etapa previa. El entendimiento de estos artefactos le permite tener ideas sobre el estado actual de la organización y como esta debe ser automatizada.

Cuando el entendimiento no es claro, gestiona reuniones de trabajo con los actores y cliente para determinar si la asociación de ideas concebida es correcta o si existen otros factores que no han sido tomados en cuenta en la etapa previa, pudiendo insertarlas en este momento.

En esta etapa se logra un conjunto de pormenorizaciones de información llegando al nivel de dato que permite la preparación de la información para la persona que va a llevar a cabo la codificación del producto, es decir, desde la especificación de requisitos de software hasta la forma de como automatizar el proceso.

Si el entendimiento es poco claro entonces debe iterar en su procedimiento hasta lograr entender que los procesos a automatizar van a cumplir su función, así como los datos necesarios que van a permitir un manejo algorítmico adecuado permitiendo una definición precisa de los datos a manejar.

El procedimiento que permite lograr el objetivo de esta etapa se muestra a continuación:

1. Recoger todas las plantillas de la etapa de educación de requisitos. Esta información se encuentra libre de inconsistencias o ambigüedades.
2. Planificar el análisis de la fase de educación de requisitos con todos los requisitos educacionados otorgando prioridades a cada requisito educacionado.
3. Analizar un requisito educacionado.
4. Definir la ilación de requisitos para cada requisito educacionado.
5. Decidir sobre la cantidad de requisitos ilados que surgen por cada requisito educacionado.
6. Analizar cada requisito ilado.
7. Si existen inconsistencias o ambigüedades recurrir al requisito educacionado.
8. Buscar las fuentes de los requisitos educacionados.
9. Planificar entrevistas de trabajo con las fuentes de los requisitos educacionados.

10. Comentar las diferencias con los entrevistados.
11. Solicitar aclaraciones.
12. Versionar los requisitos ilados hasta que no contengan inconsistencias o ambigüedades generando tantas entrevistas como sean posibles hasta encontrar un requisito ilado bien definido.

12.6 Procedimiento para corregir en la especificación

El procedimiento para corregir requisitos en esta etapa se muestra a continuación:

1. Recolectar las plantillas de ilación de requisitos las mismas que no contienen inconsistencias o ambigüedades.
2. Otorgar prioridades a estas plantillas dependiendo del conocimiento del problema.
3. Analizar el requisito especificado y de esta manera con todo el resto de los mismos.
4. Analizar cada requisito especificado.
5. Si existen inconsistencias o ambigüedades recurrir al requisito ilado para su corrección.
6. Buscar las fuentes en los requisitos ilados.
7. Iterar hasta encontrar soluciones concretas.
8. Versionar los requisitos especificados hasta que no contengan inconsistencias o ambigüedades.

12.7 Ambigüedades comunes

Los casos de uso se han convertido en una de las técnicas más populares en los métodos orientados a objetos. Ivar Jacobson los llevó a la fama y les proporcionó una serie de utilidades como una técnica informal además de darles un lugar en el desarrollo de software. Desde entonces, muchos investigadores del diseño orientado a objetos han hablado de los casos de uso. Naturalmente, estos juegan un papel importante en el Lenguaje de Modelado Unificado (UML).

Los casos de uso son valiosos por muchas razones. En primer lugar, nos ayudan a descubrir las necesidades o requisitos del sistema, sean estos en la etapa de educación, ilación o especificación de requisitos de software. Los casos de uso permiten captar las necesidades de un usuario, centrándose en una tarea que el usuario necesita hacer. La ayuda de los casos de uso también implica que podemos formular las pruebas del sistema para asegurarnos de que este se encuentra realmente integrado en el sistema. También nos ayudan a controlar el desarrollo iterativo: en cada iteración del desarrollo, se puede crear un subconjunto de los casos de uso requeridos.

A pesar de este amplio uso, hay muchos problemas sobre los casos de uso que no se conocen bien. La definición básica es bastante simple. Tan pronto como se empieza a tratar

de captar los casos de uso de un sistema, nos encontramos con preguntas sin respuesta. En la experiencia personal en el desarrollo de software se ha visto varios proyectos que se enredan y confunden con los casos de uso, generándole problemas al proceso de desarrollo del software. Una clasificación general para estas ambigüedades son los que se proponen a continuación:

12.0.1 Surgidas por la descomposición de la aplicación

Cuando se pone mucho esfuerzo en la estructuración de los casos de uso, los proyectos de desarrollo de software empiezan a tener muchas dificultades. El Lenguaje de Modelado Unificado utiliza un par de relaciones que son empleados entre los casos de uso: include y extend. Estos pueden ser muy útiles, pero causan problemas con mucha frecuencia, sobre todo la relación <include>. En tales casos, los analistas toman un caso de uso bastante general y lo dividen en sub-casos de uso. Cada sub-caso de uso se puede desglosar, por lo general, hasta llegar a algún tipo de caso de uso elemental, que parece atómica en algún grado.

Esta descomposición funcional es, un estilo de diseño que es la antítesis del desarrollo orientado a objetos. Esto conduce a dos tipos de problemas:

1. Cuando la estructura de los casos de uso se ve reflejada directamente en el código, de modo que el diseño de los casos de uso del sistema se parece a los casos de uso del usuario. Un síntoma común es encontrar al comportamiento de los objetos manipulando datos, esto es poco más que una estructura de datos encapsulada.

Este tipo de diseño ayuda a perder la mayor parte de los beneficios y características de los objetos. El sistema duplica el comportamiento a través de los distintos controladores, y el conocimiento de esta estructura de datos también es conocido por los demás controladores. La esencia del problema es que la descomposición funcional nos anima a pensar en el contexto de un comportamiento de alto nivel. Hecho de esa manera, es difícil utilizar ese mismo comportamiento en otro contexto, aunque sea casi idéntica. La descomposición funcional no promueve ese enfoque. Se puede evitar este problema, recordando que la estructura interna del sistema no tiene que parecerse a la estructura externa del mismo (los casos de uso). Sin embargo, este es un concepto difícil para los desarrolladores sin experiencia ya que son los más propensos a crear una descomposición funcional.

En la figura 12.2 se muestra como el analista comienza proporcionando un caso de uso general y en el afán de seguir entendiendo la lógica del negocio termina estructurando casos de uso que colinda con las expectativas del software perdiendo la esencia de la filosofía de los objetos.

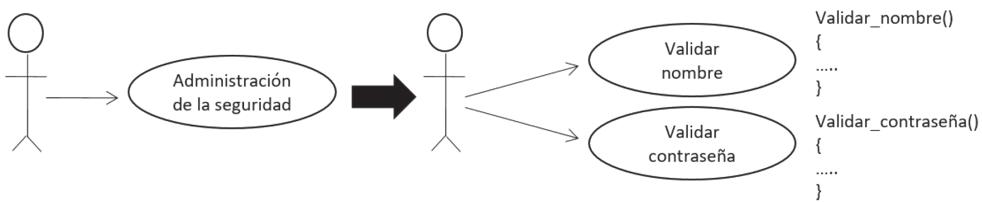


Figura 12.2: Ambigüedad por la descomposición funcional

2. Si no se encuentra que la descomposición funcional afecta a su diseño interior, un enorme conjunto estructurado de casos de uso tiene problema debido, fundamentalmente, a que la gente termina invirtiendo mucho tiempo en ellas. Llevar a todos los casos de uso hasta los pasos elementales implica que los argumentos sobre los casos de uso que caben en los casos de uso de mayor nivel consumen tiempo que podría ser empleado en otras cosas.
- La captura de todos los detalles de los casos de uso no es necesario en las primeras fases del desarrollo. Se necesita primero buscar en las zonas de alto riesgo, pero muchos de los detalles se pueden dejar para las últimas etapas del desarrollo iterativo. Esa es la filosofía del desarrollo iterativo. Esto causa desaliento al emplear la relación <include>. En la práctica se advierte similitudes entre el diseño de los casos de uso y los casos de uso del usuario, y estos son señales de advertencia. La relación <extend> causa menos problemas, aunque los investigadores poco dicen de sus deficiencias.

La figura 12.3 muestra el uso excesivo de relaciones entre los casos de uso haciendo que el diagrama de casos de uso sea difícil de interpretar.

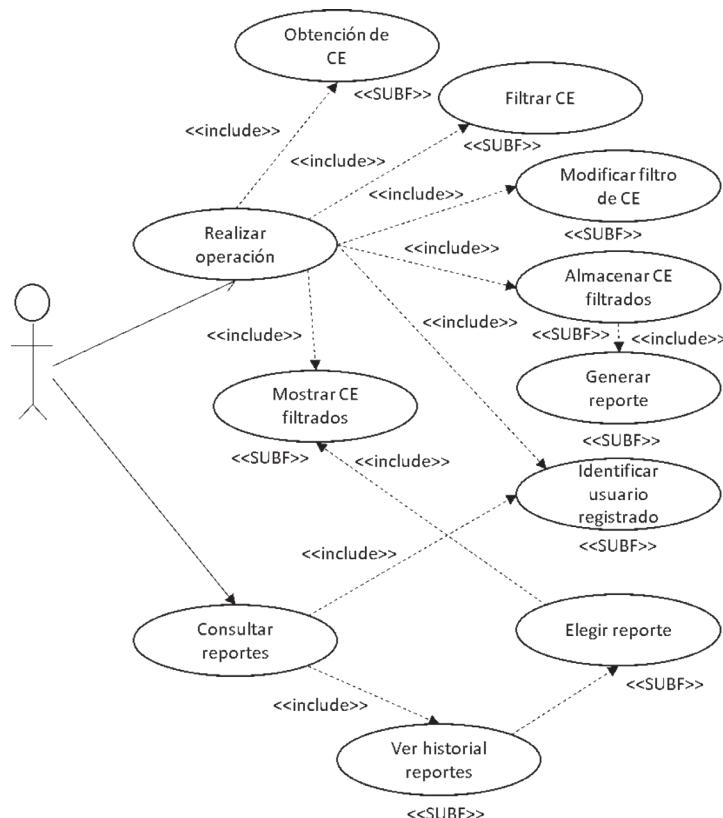


Figura 12.3: Ambigüedad por el excesivo uso del use o include

12.0.2 Surgidas por abuso de la abstracción

Los objetos nacen de la abstracción. Un buen diseño es aquel que se basa en abstracciones simples, haciendo un problema complejo manejable. Así como los diseñadores utilizan la abstracción, otros integrantes del equipo de desarrollo de software también la pueden usar.

Pero la abstracción puede llevar a problemas en los casos de uso. Comúnmente los usuarios entienden los casos de uso, al principio, y posteriormente se sienten perdidos con los casos de uso más abstractos. Uno de los principales propósitos de los casos de uso consiste en comunicarse con los usuarios – los clientes – del sistema. La abstracción de los casos de uso, más allá del nivel de comprensión, no ayudan a nadie. La falta de abstracción en los casos de uso no es perjudicial, ya que la estructura interna no tiene que ser la misma que la estructura externa.

La abstracción de los casos de uso puede conducirnos a casos de uso más grandes, que, en el desarrollo iterativo, son más difíciles de planificar. Se puede pasar mucho tiempo discutiendo acerca de la abstracción en vez de usar el tiempo en otras cosas. La figura 12.4 muestra un ejemplo clásico de la definición de una excesiva abstracción haciendo que el caso de uso no proporcione la información real del proceso.

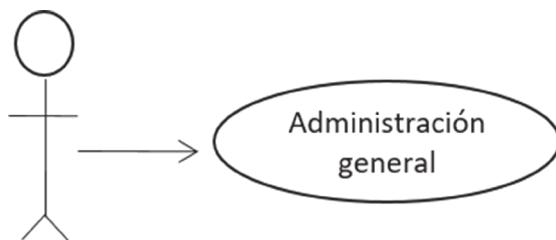


Figura 12.4: Ambigüedad por abuso de abstracción

12.0.3 Surgidas por el uso de interfaces gráficas de usuario

Con todas las herramientas para diseñar interfaces gráficas de usuario que existen en estos días, muchos desarrolladores las están usando para ayudar a determinar los casos de uso. Esta lógica es atractiva. Una interfaz gráfica de usuario es concreta a un usuario, nos ayuda a no olvidar detalles; da al usuario una sensación de cómo se verá el sistema. Las interfaces gráficas de usuario son fáciles de prototipar ya que hacen demostraciones razonables para explicar la capacidad del futuro sistema.

Existe un problema fundamental. Al mostrar un prototipo de interfaz gráfica de usuario a un usuario, parece que casi todo lo hace, y que lo único que queda son pocos detalles detrás de las escenas. Sabemos que lo que se esconde detrás de las escenas es la parte más complicada del ejercicio, pero esto es muy difícil de entender para los clientes. Las interfaces gráficas de usuario conducen a una falsa indicación de los avances y dificultades. Hay una enorme diferencia entre el esfuerzo real y el esfuerzo percibido, por lo que es difícil hacer la negociación que es tan importante en el control del alcance.

A pesar del hecho de que las herramientas para interfaces gráficas de usuario parecen tan fáciles de usar, siempre existen ajustes para que se vea perfecto. Este ajuste fino establece expectativas de la gente en una determinada dirección, haciendo que la gente se vuelva reacia a realizar cambios. La figura 12.5 muestra interfaz gráfica de usuario con todos los elementos que contiene el software real dando una impresión de ser un producto definitivo.

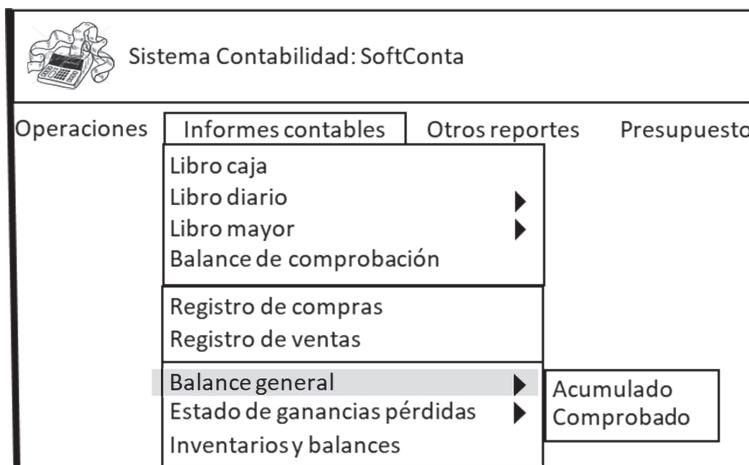


Figura 12.5: Ambigüedad por uso de la interfaz gráfica

12.0.4 Surgidas por la reutilización de plantillas

En muchas ocasiones insertamos, en el proyecto actual, plantillas de casos de uso de otros proyectos llevados a cabo. El problema con estos casos de uso es que no abordan directamente las necesidades de un usuario. Las necesidades reales del usuario son algo así como garantizar un formato consistente dentro de un documento.

El problema con ir directamente al caso de uso del sistema es que se le niega la oportunidad de llegar a los casos de uso de otro sistema que se ocupe de lo mismo. No se puede utilizar los casos de uso del usuario, debido a que no encajan bien en el proceso de programación iterativo.

En un principio se pone atención a los casos de uso del sistema, ya que son los más útiles para la planificación de la iteración y pruebas del sistema. Sin embargo, con todos los casos de uso del sistema puede existir otro caso de uso de usuario que lo respalda.

La figura 12.6 muestra un caso típico de emplear un caso de uso orientado a la seguridad de un proyecto de software sobre otro proyecto de software sin considerar los criterios propios de seguridad contemplado por la nueva organización.

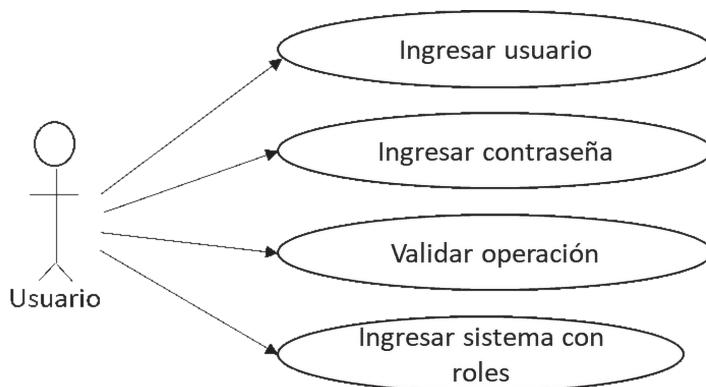


Figura 12.6: Ambigüedad por el uso de plantillas de otros proyectos

12.0.5 Surgidas por el mal diseño de los casos de uso

Cuando se planifica la construcción de un sistema automatizado se prevé que los artefactos generados en las etapas iniciales sean adecuados para la etapa de diseño. El defecto de los casos de uso, para esta etapa, implica que no existen los mecanismos adecuados para lograr determinar los comportamientos generales o especializados haciendo que el actor, relacionado con el caso de uso, viaje a través de la generalización y la especialización del caso de uso correspondiente.

Esto conlleva a que no se logre visualizar con claridad con qué tipo de caso de uso se está trabajando ya que los mismos pueden estar expresados de manera independiente o que se encuentren anidados unos a otros. Esta falta de claridad hace pensar que el diseño se encuentra bien hecho cuando en el fondo el solapamiento de casos de uso genera grandes niveles de complejidad. La figura 12.7 refleja esta situación crítica.

Normalmente se comienza diseñando casos de uso generales y a partir de ellos se definen casos de uso particulares, aunque a algunos diseñadores les gusta invertir la forma de obtenerlos. La experiencia del líder del proyecto permite ordenar la forma de trabajo y de extracción de estos artefactos.

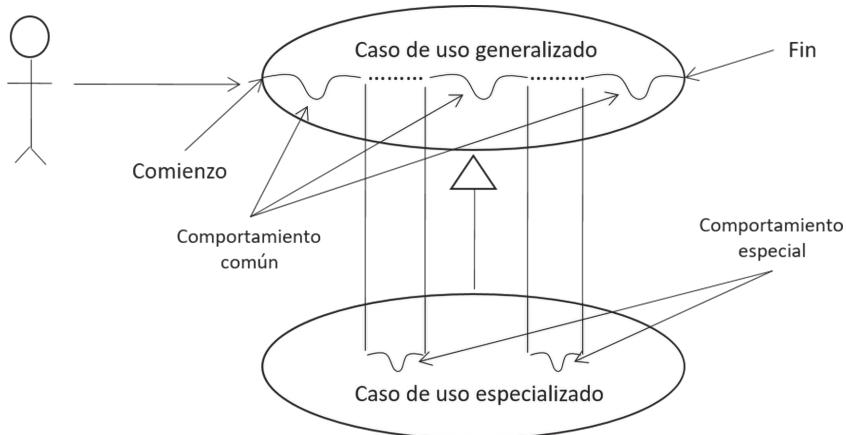


Figura 12.7: Ambigüedad por el mal diseño del caso de uso

12.0.6 Surgidas por la tecnología de objetos

Los casos de uso no son elementos que están orientados a objetos. Cada caso de uso captura una abstracción funcional importante que puede causar numerosos problemas con la descomposición funcional que la tecnología de objetos espera evitar. Esto implica que el modelo de casos de uso y el modelo de objetos pertenecen a diferentes paradigmas (es decir, funcional y orientada a objetos) y por lo tanto utilizan diferentes conceptos como en la terminología, las técnicas y notaciones.

La estructura simple del modelo de casos de uso permite no definir claramente la estructura de la red del modelo de objetos con sus objetos y las clases de colaboración. Estas asignaciones son informales y un tanto arbitrarias, sirven de poco para lograr identificar objetos, clases, y sus interacciones. El uso de include y extend no hace que tengamos una visión correcta de la complejidad.

En la figura 12.8 se puede notar que los casos de uso: "Generar Libro Caja General", "Generar Libro Diario", "Generar Registro de Compras", "Generar Registro de Ventas", "Generar Libro Mayor" y "Generar Balance de Comprobación" no pueden ser tratados como objetos ya que no significa que encapsulen datos sino escenarios conjuntamente con sus episodios.

En el hecho del caso de uso "Generar Libro Caja General" conlleva un conjunto de ítems que guardan una relación intrínseca con un proceso y por ende no presenta la abstracción necesaria para determinar los objetos, las clases, sus atributos conjuntamente con sus asociaciones. Solo el verbo "Generar" implica un acto que no guarda asociación con objeto alguno por lo que resulta imposible deducirlo.

Puede que la palabra "Libro" nos lleve a una confusión ya que el mismo lo podemos tratar como

un objeto físico asociado con una abstracción propia; pero la primera palabra lo convierte en un proceso que se encuentra asociado con un escenario específico y que dentro del mismo se presentan un conjunto de episodios que en conjunto genera la acción del proceso y su interrelación con hechos objetivos. Lo más resaltante de este problema es que nos permite determinar de manera clara los escenarios y episodios que se encuentran inmersos en la solución pero que se presentan de manera engañosa cuando de objetos y clases se trata.

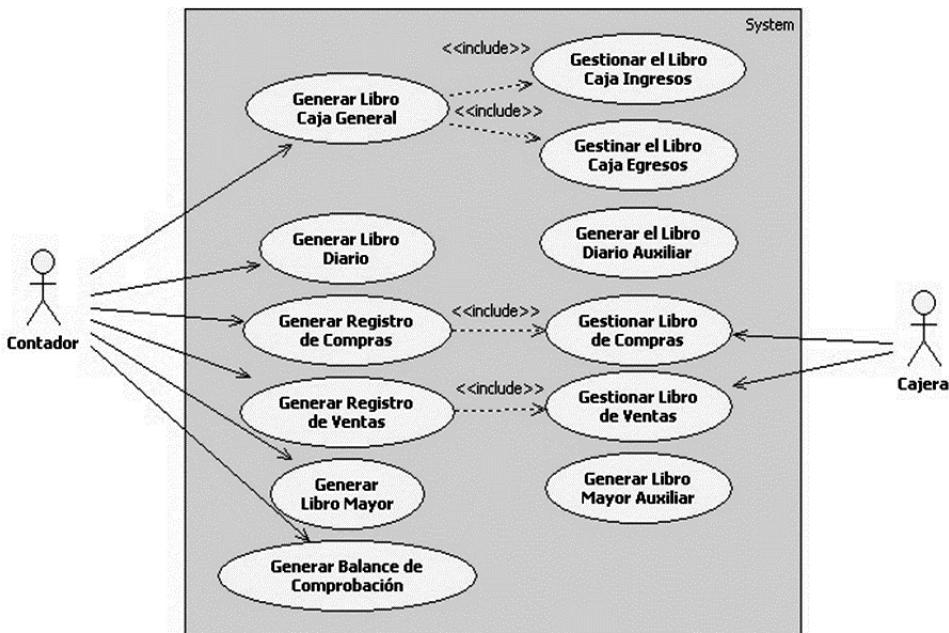


Figura 12.8: Ambigüedad surgida por la tecnología de objetos

12.0.7 Surgidas por las iteraciones

Otro problema potencial con el modelado de casos de uso es no saber cuándo parar. Cuando uno está construyendo una aplicación no trivial, a menudo hay un gran número de casos de uso que pueden producir un número esencialmente infinito de escenarios de uso, especialmente con interfaces gráficas de usuario.

En la figura 12.9 se muestra la primera concepción que se tuvo para desarrollar el producto de software. El diagrama de caso de uso no explica mucho sobre lo que se tiene que construir; conforme se avanzó en el proyecto, se necesitaba de una mayor especificación por lo que se refinaba el diagrama de casos de uso con mayor detalle (figura 12.10).

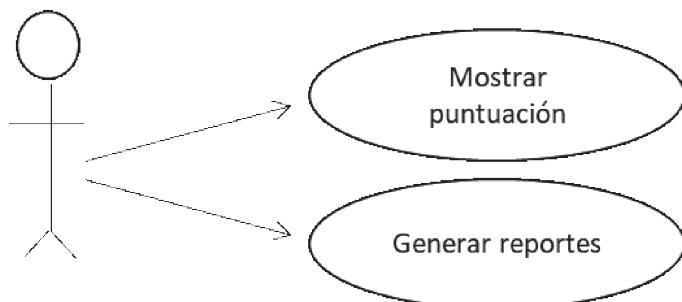


Figura 12.9: Ambigüedad surgida por las iteraciones - inicial

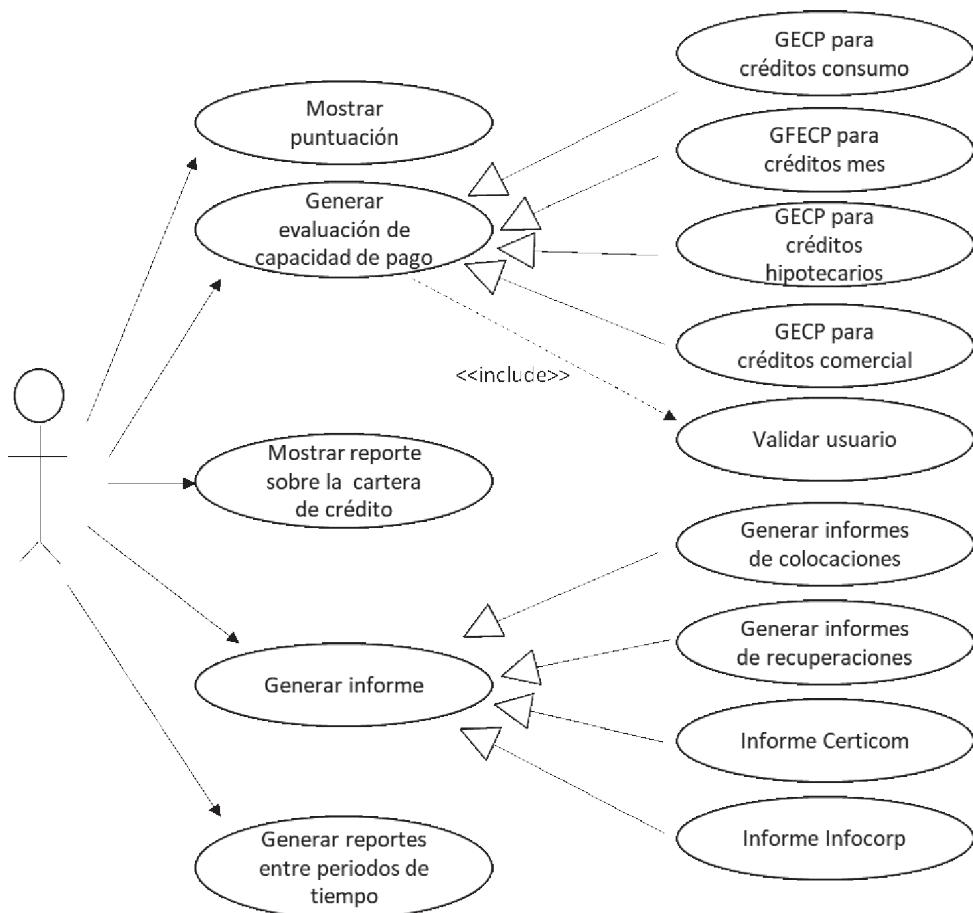


Figura 12.10: Ambigüedad surgida por las iteraciones - final

12.0.8 Surgidas por la arquitectura

Otro problema importante corresponde a la arquitectura del sistema que puede dar lugar a una mala interpretación de los casos de uso, debido a la mala interpretación que se lleva a cabo de la arquitectura del software. Dichas arquitecturas, típicamente, exhiben encapsulación pobre y de acoplamiento excesivo, y una inadecuada distribución de la inteligencia de la demanda entre las clases.

La figura 12.11 muestra la arquitectura del sistema automatizado CoopSOFT, y de ella se puede interpretar que el Navegador Web, en la capa del cliente, puede tener un comportamiento orientado al caso de uso. Esta errónea interpretación nos conduce a suponer que de ella se pueden obtener un conjunto de clases que se encuentran orientadas a resolver la demanda de los servicios.

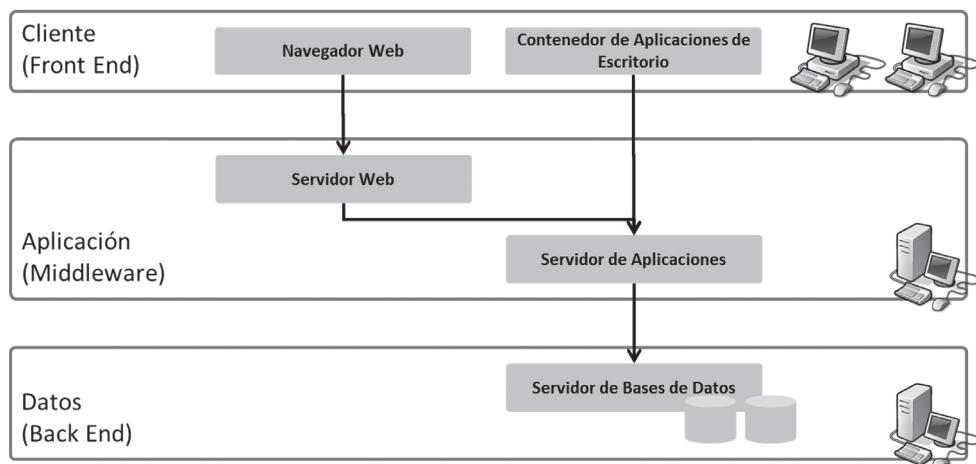


Figura 12.11: Ambigüedad surgida por la arquitectura

12.0.9 Surgida por el uso de escenarios

Esta ambigüedad surge por la poca claridad que existe entre el concepto de caso de uso y escenario. La figura 12.12 muestra que en una de las primeras versiones para conceptualizar los casos de uso se crearon los casos de uso llamados: “Enviar estado de cuenta al cliente con préstamo” y “Enviar estado de cuenta al cliente moroso”. En realidad, estos son los escenarios del caso de uso padre denominado: “Enviar estado de cuenta”.

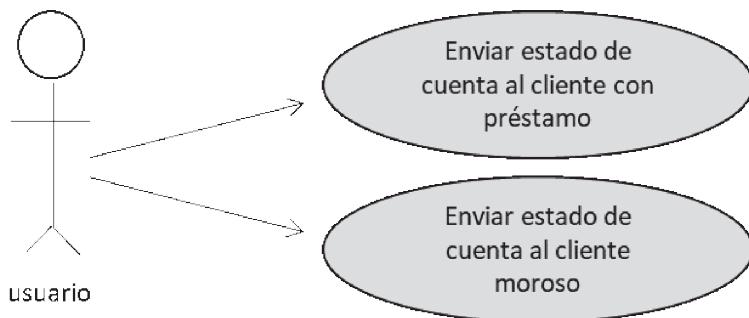


Figura 12.12: Ambigüedad surgida por los escenarios

12.0.10 Mala interpretación de los dominios del sistema

Este problema ocurre cuando se desconocen los dominios del sistema real a construir o cuando no se tiene muy en claro las áreas o módulos que contiene el mencionado sistema. Generalmente los analistas tratan de representar, al mismo tiempo, a los usuarios del sistema y a los usuarios del negocio. En la figura 12.13 se muestra como el analista inserta en el diagrama de casos de uso al cliente (socio) y al usuario del sistema (operador). Dos elementos totalmente disjuntos para el sistema real.

La literatura relacionada con el tema generalmente no explica el problema del dominio y cuando los ingenieros de software novatos se enfrentan a esta realidad no saben como resolver el problema. Esto se debe a que no existe una forma de detectar los componentes principales del dominio del sistema o por lo menos descubrir algunas iniciativas que concuerden con este hecho.

Una forma de detectar la cantidad de áreas o módulos que contiene un sistema automatizado es llevar a cabo las educciones del caso; después de refinar y versionar las mismas nos encontramos con una cantidad finita de educciones, las mismas que concuerdan con la cantidad de módulos a desarrollar dentro del sistema a automatizar.

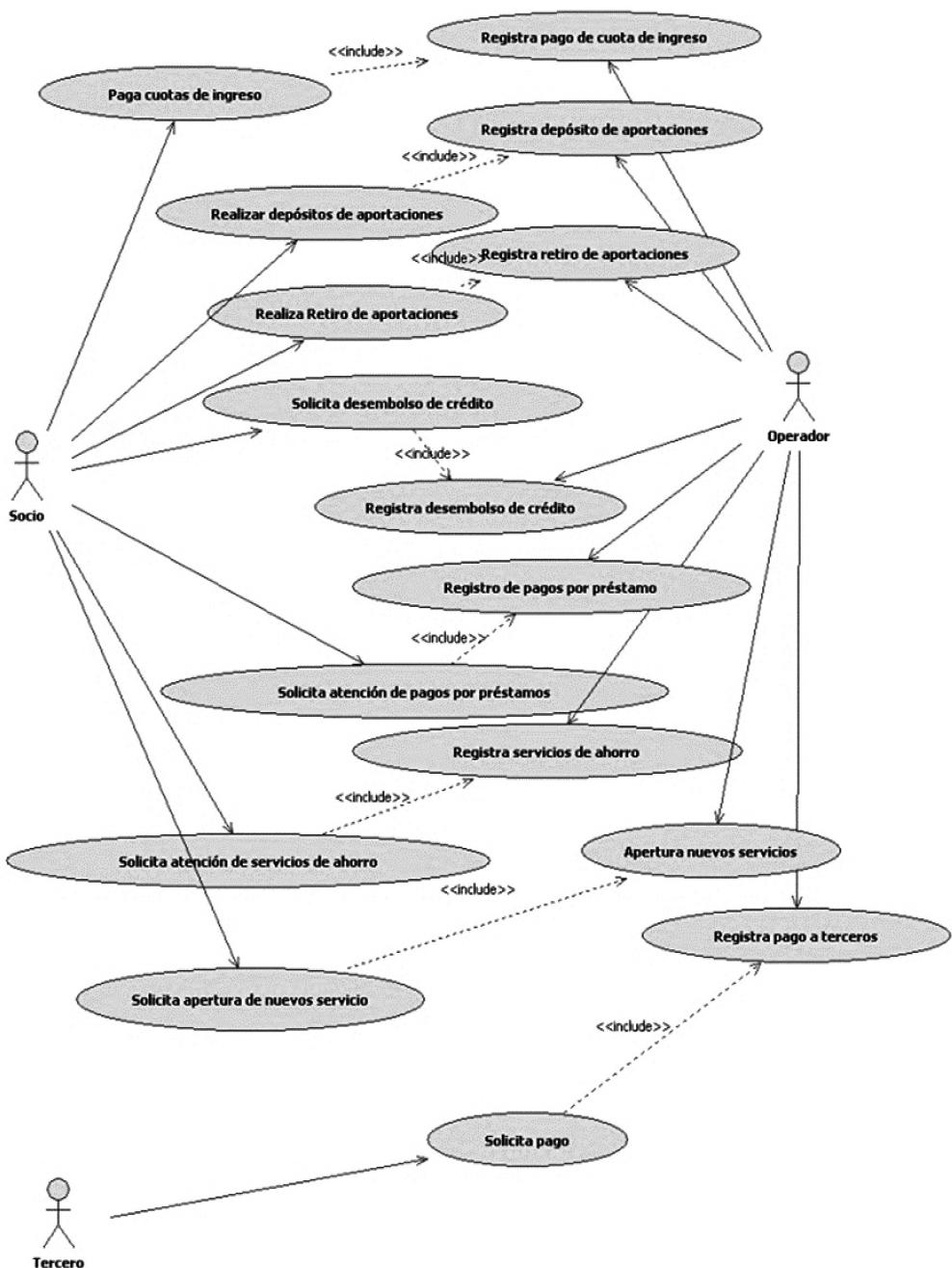


Figura 12.13: Ambigüedad surgida por deficiencia en los dominios

12.0.11 Surgidas por la educación de requisitos

Esta es la primera etapa de la ingeniería de requerimientos. En ella se toma contacto con el cliente para empezar a entender el problema que se tiene que resolver. Generalmente los clientes no entienden de computación o informática y proporcionan sus ideas de manera desordenada pensando que el analista puede entenderlas a cabalidad.

Esta percepción lleva a una mala interpretación de los casos de uso ya que, en la generalidad, los analistas empiezan a esbozar los primeros casos de uso sobre problemas técnicos y no sobre la problemática de los procesos; y en muchos casos confundiendo los objetivos de la construcción del software con la definición de los casos de uso (tabla 12.1).

Atributos	Descripción
EDU-0005	Negociación de Crédito
Versión	1.0 (22/06/2006)
Autor	Manuel Mayoría Salas
Fuente	Maritza Paz Calcín
Descripción	El sistema deberá lograr que la negociación del crédito considere la situación del socio en el Sistema Financiero y éste se otorgue respetando la necesidad, el encaje y la capacidad económica del socio, para no perjudicarlo en el futuro.
Importancia	Importante
Urgencia	Hay presión
Estado	Pendiente de verificación
Estabilidad	Media
Comentarios	Ninguno

Tabla 12.1: Ambigüedades surgidas por la educación de requisitos

Al establecer las cualidades para un sistema, es importante identificar todas las categorías de usuarios (incluyendo de otros sistemas) que interactuarán con el sistema, y entender que atributos de calidad se debe tener en cuenta. Un atributo de calidad como el desempeño puede surgir para un usuario como un interés, y para otro como un valor, por eso es útil una educación directa de requisitos para ambos, es decir interés y valor para cualquier (grupo) de usuario(s). Es importante dirigir a crear “*justo lo que quieren los usuarios*”, con las cualidades que ellos toman en cuenta; los rasgos de baja prioridad o cualidades solo incrementan la complejidad para la organización de desarrollo y/o el usuario.

El equipo de requerimientos debería, sin embargo, estar alerta a requerimientos que el usuario presupone o que no estén disponibles para articularse directamente. Entender el objetivo de los usuarios y forzar a que impacte su suceso y sentido de utilidad, ayudará a surgir y establecer las prioridades de cualidades del sistema, así como la funcionalidad. En adición a descubrir qué cualidades son importantes al usuario en el nivel de sistema, las

cualidades asociadas a una función particular u objetivo de usuario debe ser educido. Las cualidades pueden necesitar ser trasladadas por los desarrolladores desde los objetivos de nivel usuario, valores e intereses, en requerimientos específicos de calidad técnica. Por ejemplo, un requerimiento de usuario no puede ser degradado o impedido por el rendimiento lento del sistema, al realizar una tarea que puede ser trasladada en requerimientos sobre el tiempo de transacciones y la potencia de red.

12.0.12 Surgidas por la ilación de requisitos

Esta es la segunda etapa de la ingeniería de requerimientos. El ingeniero de software tiene en mente toda la problemática del problema a resolver; incluso se arriesga a representar el problema con los primeros casos de uso. Después de varias iteraciones concibe el documento de ilación de requisitos. A partir de este artefacto se diseñan los casos de uso posteriores.

Las fallas de redacción, ortografía, sintaxis y semántica conllevan a una mala interpretación de los casos de uso. El principal resultado del proceso de ilación de requisitos es la serie de requisitos que deberán ser utilizados por el equipo de desarrollo de software para crear un producto correcto y de manera apropiada. Además de este resultado principal, el proceso ofrece una serie de salidas intangibles.

Cuando existe un buen proceso de ilación de requisitos, se ayuda a los usuarios a entender qué es lo que quieren, qué es lo que necesitan, cuáles son las restricciones y alternativas, ventajas y desventajas de cada una. Desde el punto de vista de los ingenieros y desarrolladores, la existencia de un buen proceso de ilación de requisitos ayuda a los mismos a resolver el problema correcto, es decir, lo que realmente desea y necesita el cliente, a resolver un problema factible, a ganar la confianza del cliente y su cooperación y a ganar conocimiento sobre el dominio del problema.

Pero ¿cuáles son las inconsistencias y ambigüedades que inserta el proceso de ilación de requisitos? Un resumen se muestra a continuación:

1. El mal entendimiento del modelo del negocio hace que las ideas no queden claras y por lo tanto el ingeniero suministre información de acuerdo a su percepción.
2. La duplicidad de conceptos en diferentes tablas de ilación permite introducir errores en el momento de la implementación del producto.
3. La mala trazabilidad permite que sólo se corrijan aquellos errores que se pueden ver dejando de lado aquellos que se encuentran ocultos en otras tablas, ya sea de educación o de especificación.
4. La mala interpretación de la información que se encuentra en la tabla producto de la mala redacción permite el ocultamiento del verdadero criterio a tomar en cuenta.
5. Una pésima codificación, permitiendo una mala relación con las otras tablas, hace que se

corrija a un grupo de ellas dejando de lado al otro.

6. Una mala especificación de las tablas asociadas permite que se introduzcan errores en aquellas tablas que probablemente se encuentren correctas.
7. Una mala redacción de las actas de entrevistas permite malas interpretaciones e introducen errores colaterales en el resto de las tablas.
8. Una mala corrección de las tablas permite que se introduzcan nuevos errores en lugar de eliminarlos.
9. Una mala asociación con artefactos de diseño permite que se corrijan los defectos en las tablas más no en los artefactos asociados.

Estas inconsistencias o ambigüedades pueden generar tantas incongruencias en las tablas de ilación y su asociación con las tablas de educación y especificación que afecta en la codificación del producto de software. Incluso va a permitir que los codificadores inserten criterios personales de codificación que no estaban previstas o planificadas por los ingenieros de software. Estos problemas generan peligrosidad al momento de la codificación ya que genera una serie de incertidumbres cuando, por ejemplo, de estructura de datos se refiere.

La solución se encuentra en hacer correcciones de manera ordenada y escalonada; por ejemplo, primero corregir las tablas de educación hasta estar seguros de ello y posteriormente las tablas de especificación y finalmente todo artefacto de diseño asociado. Corregida una tabla y su relación se procede a corregir los diagramas asociados y con criterios de control de versiones. Los diagramas también son versionados y nunca dejados de lado ya que de ellos se desprende los criterios de codificación correspondientes.

Cuando el proceso de ilación es pobre es posible que se dé respuesta a un problema equivocado (tabla 12.2), los usuarios podrían expresar su descontento ya que los desarrolladores no los escuchan y se produciría un desarrollo caótico en el sentido que se pierde información, se toman decisiones incorrectas, los costos y el cronograma se salen del plan y se pierde dinero. La alternativa de producto de software que se desarrolla podría afectar el proceso de ilación ya que el contacto con el usuario y el conocimiento del mismo surgen desde medios diferentes según se desarrolle software a medida o empaquetado. Los usuarios para una u otra alternativa de software tienen diferentes expectativas y, muy probablemente, diferente percepción del proceso de ilación de requisitos.

Atributos	Descripción
ILA-0002	Almacenar descripción de créditos aprobados
Versión	1.0 (22/06/2006)
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Dependencias	<p>Las dependencias son las siguientes:</p> <ul style="list-style-type: none"> [EDU-0003] Mostrar reportes sobre la cartera del cliente. [EDU-0004] Generar informes de colocaciones. [EDU-0006] Generar reportes entre períodos de tiempo establecidos por el usuario. [EDU-0007] Generar plan de pagos. [EDU-0008] Generar reporte de morosidad de plan de pagos. [EDU-0009] Generar reporte de evaluación de cartera de crédito. [EDU-0026] Mostrar crédito aceptado. [EDU-0029] Mostrar relación de créditos por convenio. [EDU-0030] Mostrar número de créditos para descuento de socio por convenio. [EDU-0031] Calcular monto total de créditos para descuento de socio por convenio. [EDU-0036] Mostrar créditos de acuerdo con su tipo. [EDU-0060] Relación del historial de socios con mora. [EDU-0061] Reporte de socios con préstamo presente. [EDU-0044] Atención de pagos por préstamo. [EDU-0045] Desembolso de créditos. [EDU-0052] Impresión de órdenes de pago por crédito.
Descripción	El sistema deberá almacenar la información correspondiente a la descripción de créditos aprobados por la cooperativa.
Datos específicos	Ninguno
Tiempo de vida	Tres días
Ocurrencias simultáneas	Por definir
Importancia	Importante
Urgencia	Hay presión
Estado	Pendiente de verificación
Estabilidad	Media
Comentarios	Ninguno

Tabla 12.2: Ambigüedades surgidas por la ilación de requisitos

La experiencia de los analistas cobra real importancia. Si no han trabajado sobre productos anteriores entonces existe una alta probabilidad de introducir errores y uno común es distorsionar el modelo del negocio. Esta distorsión también implica una mala codificación y finalmente una mala construcción del producto final.

12.0.13 Surgidas por la especificación de requisitos

Esta es la tercera etapa de la ingeniería de requerimientos. El ingeniero de software transfiere todo su conocimiento en esquemas orientados a la codificación; realiza transformaciones que se encuentran orientadas a definir las estructuras en criterios de programación como lo son: estructuras de datos, tipos de datos, entre otros. Después de un conjunto de iteraciones concibe el documento de especificación de requisitos. El problema se encuentra en la

minuciosidad del código; es común trabajar sobre seudocódigo como aspecto genérico.

La mala redacción puede introducir problemas de inconsistencias y ambigüedades incluido una mala definición de datos. Como resultado se obtiene un conjunto de requisitos orientados a la codificación. Cuando este proceso es pobre es posible que se obtengan respuestas inadecuadas tal como se muestra en la tabla 12.3. Se puede notar que en el atributo “Descripción”, su contenido “*Vaciar estructura de memoria interna al disco duro*” no indica si el vaciado lo debe de hacer utilizando árboles, hilos, tablas hash, montículos binarios, listas enlazadas, almacenamiento basado en bloques, entre otros.

Atributos	Descripción
ESP-0008	Almacenar descripción de créditos aprobados
Versión	1.3 (22/06/2006)
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Dependencias	Las dependencias son las siguientes: [ILA-0006] Almacenar información en memoria principal. [ILA-0007] Generar información de créditos.
Descripción	Vaciar estructura de memoria interna al disco duro.
Datos específicos	Ninguno
Importancia	Importante
Estabilidad	Media
Comentarios	Ninguno

Tabla 12.3: Ambigüedades surgidas por la especificación de requisitos

12.8 Análisis de inconsistencias y ambigüedades

Se establece el procedimiento para eliminar inconsistencias o ambigüedades que se pueden presentar tanto en la etapa de educación de requisitos como en la etapa de ilación y especificación. Este procedimiento implica que los artefactos deben ser versionados para llevar un control adecuado de las correcciones por medio de la trazabilidad de las plantillas de trabajo. El procedimiento es el siguiente:

1. Preparar las plantillas de los requisitos educacionados e ilados.
2. Proporcionar un orden a las plantillas de educación y su relación o asociación con las plantillas de ilación.
3. Comenzar por la plantilla de educación, y su asociación con las plantillas de ilación, que se consideren el punto de partida del sistema a construir.
4. Por cada plantilla de educación y su respectiva asociación con las plantillas de ilación, aplicar la resolución de inconsistencias y ambigüedades.
5. Analizar las plantillas de requisitos en forma individual, así como las plantillas de ilación relacionadas.

6. Documentar las inconsistencias o ambigüedades encontradas en las plantillas de trabajo.
7. Planificar las entrevistas con las fuentes que otorgaron la información y que se encuentran en cada plantilla de educación e ilación.
8. Conversar con la fuente que entregó el requisito educacionado o ilado.
9. Interpretar las ideas en las fases correspondientes.
10. Preparar un prototipo demostrativo sobre los diseños del producto y asociados con los requisitos educacionados o ilados.
11. Mostrar y corregir el prototipo, en caso fuera necesario, retroalimentando las plantillas de educación e ilación con criterios de control de versiones.
12. Revisar los cambios efectuados producto del control de versiones hasta encontrar la satisfacción del analista.

12.9 Solución a las inconsistencias y ambigüedades

Se presenta el procedimiento para eliminar inconsistencias o ambigüedades que se puedan presentar tanto en la educación, ilación o especificación de requisitos. Este procedimiento implica que los artefactos deben ser versionados para llevar un control adecuado de las correcciones por medio de la trazabilidad de plantillas de trabajo. El procedimiento es el siguiente:

1. Preparar las plantillas de los requisitos educacionados, ilados y especificados.
2. Proporcionar un orden a las plantillas de educación y su relación o asociación con las plantillas de ilación.
3. Proporcionar un orden a las plantillas de ilación y su relación o asociación con las plantillas de especificación.
4. Comenzar por la plantilla de educación, y su asociación con las plantillas de ilación, que se consideren el punto de partida del sistema a construir.
5. Por cada plantilla de educación y su respectiva asociación con las plantillas de ilación, aplicar la resolución de inconsistencias y ambigüedades.
6. Analizar las plantillas de requisitos en forma individual, así como las plantillas de ilación relacionadas.
7. Hacer lo mismo entre las plantillas de ilación y especificación.
8. Documentar las inconsistencias o ambigüedades encontradas en las plantillas de trabajo.
9. Planificar las entrevistas con las fuentes que otorgaron la información y que se encuentran en cada plantilla de educación, ilación y especificación.
10. Conversar con la fuente que entregó el requisito educacionado, ilado o especificado.
11. Interpretar las ideas en las fases correspondientes.
12. Preparar un prototipo demostrativo sobre los diseños del producto y asociados con los requisitos educacionados, ilados o especificados.

13. Mostrar y corregir el prototipo, en caso fuera necesario, retroalimentando las plantillas de educación, ilación y especificación con criterios de control de versiones.
14. Revisar los cambios efectuados producto del control de versiones hasta encontrar la satisfacción del analista.

12.10 Confección del catálogo de requisitos

Esto consiste en establecer el documento oficial denominado catálogo de requisitos el mismo que debe ser transmitido al área de desarrollo de software para su respectiva codificación en el lenguaje de programación previsto en los requisitos no funcionales y bajo las condicionalidades de la infraestructura del cliente.

El catálogo de requisitos es un artefacto que se encuentra libre de inconsistencias y ambigüedades y que se encuentra conformado por los artefactos de educación, ilación y especificación. Este documento también va a permitir lograr una trazabilidad de las necesidades del cliente hasta el prototipo construido para su comprobación.

El procedimiento para determinar el catálogo de requisitos, que también debe ser versionado, es el siguiente:

1. Entregar formalmente el catálogo de requisitos al responsable del área de desarrollo de software. Esta documentación es importante por lo que la entrega debe ser formal y su consignación adecuada.
2. Definir los criterios para su respectiva codificación respetando prioridades y asociaciones. Incluso deben definirse las formas de codificación y el estándar a seguir; lo que se busca es un código limpio en su entendimiento.
3. Separar las plantillas de requisitos dependiendo del análisis de modularidad fijado para el producto final. También las plantillas pueden ser adecuadas producto de la experiencia en la construcción de otros aplicativos.
4. Codificar el producto bajo los criterios solicitados en las plantillas de trabajo. Los codificadores deben deducir los elementos de codificación desde los artefactos construidos y no con criterios personales.
5. Preparar el prototipo, en ambiente de programación, de lo solicitado en las plantillas de trabajo y tomando en cuenta todas explicaciones entregadas por el cliente y por los usuarios finales.
6. Si existe una diferencia en las visiones de trabajo entonces enviar mensaje al analista para su absolución; caso contrario aplicar el procedimiento previstos para corregir las inconsistencias y ambigüedades.

12.11 Ejemplo del proceso de corrección

12.0.14 Ejemplo de inconsistencia

Veamos el caso de la plantilla de educación (tabla 12.4) en donde se muestra la inconsistencia producto de que en la *descripción* no existe una especificación coherente y en el campo *comentarios* no especifican los tipos de usuarios. Esto es producto de varios factores como la inexperiencia del ingeniero, el mal entendimiento de la información, la mala comprensión del modelo del negocio, entre otros.

Atributos	Descripción
EDU-0004	Gestionar paquete
Versión	1.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema debe permitir operar con destinos turísticos los mismos que son ofertados.
Importancia	Vital
Estado	Inicial
Comentarios	Usada por usuarios.

Tabla 12.4: Plantilla de educación - Primera iteración

Para corregir estas inconsistencias se pueden generar varias iteraciones hasta corregir las mismas. Incluso, al llevar a cabo las correcciones se pueden crear más tablas de educación. No cabe duda de que el control de versiones también puede introducir inconsistencias en las nuevas tablas, pero es la experiencia del analista la que ponderará los nuevos requisitos eligiendo aquellos que presentan inconsistencias y ambigüedades. Las correcciones se muestran en la tabla 12.5.

Atributos	Descripción
EDU-0004	Gestionar paquete
Versión	2.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema permitirá: Agregar paquete, eliminar paquete, modificar paquete, crear paquete, reservar paquete y cancelar paquete.
Importancia	Vital
Estado	Inicial
Comentarios	Los usuarios serán: Secretaria, administrador.

Tabla 12.5: Plantilla de educación - Segunda iteración

Aun así, las inconsistencias se encuentran presentes debido al significado de paquete en forma explícita. Un paquete puede tratarse de un objeto físico que puede contener algún objeto, o conjunto de objetos, en su interior; o puede tratarse de un paquete turístico como por ejemplo la oferta de un conjunto de servicios ofrecidos a los turistas. La tabla 12.6 muestra la tercera iteración.

Es probable que todas las inconsistencias no sean detectadas o que al hacer las correcciones se introduzcan otras. Por ejemplo, en la tabla 12.5 es probable que el ingeniero haya obviado alguna actividad en la descripción pensando que la misma se encuentre inmersa dentro de otra. Por ejemplo, puede suceder que “*Eliminar paquete*” y “*Cancelar paquete*” sean entendidos de la misma manera ya que no existe una descripción clara de lo que se desea hacer.

La tabla 12.6 muestra la tercera iteración en donde se puede notar una mayor especificidad; pero aun así para otros ingenieros puede suceder que todavía no existe una buena especificidad por lo que pueden proceder a solicitar una nueva iteración. Este proceso iterativo, aunque ocupa tiempo, resulta ser adecuado para el control de versiones y para detectar las inconsistencias y ambigüedades en los requisitos.

Notar que en los comentarios tampoco existe una buena especificación, no se especifica al tipo de secretaria o administrador. Se debe recordar que en los requisitos no se puede ni se debe de sobreentender nada; la especificación debe ser clara y contundente de tal forma que a las personas que los lean lo entiendan en una primera oportunidad.

Atributos	Descripción
EDU-0004	Gestionar paquete turístico
Versión	3.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcin
Descripción	El sistema permitirá: Crear paquete turístico, visualizar paquete turístico, modificar paquete turístico, eliminar paquete turístico y buscar paquete turístico.
Importancia	Vital
Estado	Inicial
Comentarios	Los usuarios serán: Secretaria, administrador. Cada paquete contiene el tour que se va a realizar.

Tabla 12.6: Plantilla de educación - Tercera iteración

12.0.15 Ejemplo de ambigüedad

En este caso se muestran dos plantillas de educación que presentan la misma codificación (tabla 12.7 y 12.8) pero que tienen contenidos diferentes. Esta ambigüedad debe ser corregida reordenando las plantillas y logrando una codificación mucho más estable. Existe la posibilidad que en el proceso de corrección se puedan fusionar o independizar tablas lo que conduce a

que los artefactos asociados también deban ser corregidos.

El hecho de que existan dos tablas con la misma codificación puede deberse a la inexperiencia del ingeniero, a la poca atención que se pone en la documentación de requisitos o a factores externos. Ante estos hechos se debe llevar a cabo el análisis correspondiente para salvaguardar la información. Puede notarse que, a pesar de tener la misma codificación, la definición de la otra tabla presenta un concepto diferente.

Este hecho obliga a la separación inmediata y a la nueva codificación de las tablas; aun así, se debe de realizar un análisis al interior de las tablas para determinar la presencia de inconsistencias o ambigüedades. Este análisis puede producir que la separación de tablas oblique a revisar los artefactos de diseño asociados y si existen diferencias hacer las correcciones adecuadas.

Por ejemplo, los comentarios de la tabla 12.8 no son claros porque no se logra ver una clara definición de los conceptos asociados ya que puede suceder que al interior de un hotel pueda existir un restaurante. Asimismo, no queda bastante clara la definición de “Baños termales”.

Atributos	Descripción
EDU-0004	Gestionar paquete
Versión	1.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema debe permitir operar con destinos turísticos los mismos que son ofertados.
Importancia	Vital
Estado	Inicial
Comentarios	Usada por usuarios.

Tabla 12.7: Plantilla de educación 1 - Con ambigüedad

Atributos	Descripción
EDU-0004	Mostrar servicios turísticos.
Versión	1.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	Se debe mostrar información sobre todos los paquetes y servicios ofrecidos.
Importancia	Vital
Estado	Ninguno
Comentarios	Como servicios se encuentran: Hotel, restaurante, baños termales.

Tabla 12.8: Plantilla de educación 2 - Sin ambigüedad

La solución a este problema consiste en recodificar las plantillas de educación de tal forma que una de ellas lleve el código EDU-0004 y la otra el código EDU-0005. Otra forma de presentarse

la ambigüedad se muestra en la tabla 12.9.

Atributos	Descripción
EDU-0004	Gestionar paquete
Versión	1.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema debe permitir operar con destinos turísticos y sus horarios.
Importancia	Vital
Estado	Inicial
Comentarios	Usada por usuarios.

Tabla 12.9: Plantilla de educación con ambigüedad interna

En esta plantilla de educación no queda claro el requisito que va a ser albergado: los destinos turísticos o los horarios. Esta ambigüedad debe ser resuelta generando dos plantillas de educación alternativas (tabla 12.10 y 12.11).

Atributos	Descripción
EDU-0004	Gestionar paquete turístico
Versión	3.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema permitirá: Crear paquete turístico, visualizar paquete turístico, modificar paquete turístico, eliminar paquete turístico y buscar paquete turístico.
Importancia	Vital
Estado	Inicial
Comentarios	Los usuarios serán: Secretaria, administrador. Cada paquete contiene el tour que se va a realizar.

Tabla 12.10: Plantilla de educación independiente 1 - Sin ambigüedad

Atributos	Descripción
EDU-0005	Gestionar paquete por horario de personal
Versión	1.0
Autor	Fabián Chuquitaype Zúñiga
Fuente	Maritza Paz Calcín
Descripción	El sistema permitirá: Crear horario del personal, visualizar horario del personal, modificar horario del personal y eliminar horario del personal.
Importancia	Vital
Estado	Inicial
Comentarios	En los horarios se visualizarán los tours que tienen asignados los conductores y guías.

Tabla 12.11: Plantilla de educación independiente 2 - Sin ambigüedad

Bibliografía

1. S. Group, CHAOS Chronicle 2003 Report, West Yarmouth. MA: The Standish Group International, 2002.
2. I. Dávila, "La evolución de la ingeniería de requisitos en el desarrollo global de software". Ventana Informática, pp. 41-56, jun 2011. Colombia.
3. RAE, Diccionario de la Real Academia Española. Asociación de Academias de la Lengua Española, 2018. 23 edición. Edición del Tricentenario.
4. IEEE, "Ieee standard glossary of software engineering terminology," IEEE Standard Board, vol. IEEE Standards Documents, sep 1990.
5. P. Zave, "Classification of research efforts in requirements engineering," ACM Computing Surveys (CSUR), vol. 4, no. 29, pp. 315-321, 1997.
6. P. Loucopoulos and V. Karakostas, System Requirements Engineering. McGraw-Hill, first ed., 1995.
7. R. Thayer, S. Bailin, and M. Dorfman, Software Requirements Engineering. IEEE Computer Society Press, second ed., 1997.
8. O. Hurtado, Modelo de Requisitos Orientado al Reuso Efectivo (MORORE). Tesis doctoral, Universidad Carlos III de Madrid, 2009.
9. I. Sommerville, Ingeniería de Software. Addison - Wesley, ninth ed., 2011.
10. R. Pressman, Ingeniería de Software. Un enfoque Práctico. McGraw.Hill, seventh ed., 2010.
11. K. Pohl and C. Rupp, Requirements Engineering Fundamentals. rockynook, second ed., 2015.
12. IEEE, Swebok v3.0 Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Press, 2014.
13. R. Young, The Requirements Engineering Handbook. Artech House, first ed., 2004.
14. K. Wiegers, More about Software Requirements: Thorny Issues and Practical Advice. Microsoft Press, first ed., 2006.
15. K. Wiegers and J. Beatty, Software Requirements. Microsoft Press, third ed., 2013.
16. M. Arias, "La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software," Revista InterSedes, vol. VI, jul 2007.
17. R. Hernández, "Control de cambios de requerimientos de software." Instituto Tecnológico de Costa Rica. Escuela de Ingeniería en Computación, jul 2012.
18. A. Duran, Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de

- Información. PhD thesis, Universidad de Sevilla, sep 2000.
19. M. Callejas, L. Castillo, y R. Fernández, "Heler: Una herramienta para la ingeniería de requisitos automatizada", Sistemas de Computación, vol. 6, jul 2010.
 20. IBM, "Ibm rational requisitepro." https://www.ibm.com/developerworks/ssa/library/IBM_Rational_feb_2008.
 21. J. Horkoff, Y. Yu, and E. Yu, "Openome: An open-source goal and agent-oriented model drawing and analysis tool", CEUR Proceedings of the 5th International i* Workshop, vol. 766, pp. 154–156, aug 2011.
 22. A. Smith, "Open source requirements management tool". <https://github.com/osrmt/osrmt>, jul 2019.
 23. S. Supakkul and L. Chung, "The re-tools: A multi-notational requirements modeling toolkit," 20th IEEE International Requirements Engineering Conference, sep 2012.
 24. Visure, "Visure requirements," jun 2017. [25] MathWorks, "Ibm rational door", sep 2004.
 25. D. Systemes, "Reqtify" Internal Document, aug 2015. [27] J. Software, "Jama." www.jamasoftware.com, dec 2010. [28] V. R. Resource, "Gatherspace," may 2006.
 26. EcuRed, "Magicdraw", may 2016.
 27. B. Macdonald, "Definición de perfiles en herramientas de gestión de requisitos". Facultad de Informática. Universidad Politécnica de Madrid, sep 2005.
 28. M. Solarte, "Amirst: Propuesta de una aproximación metodológica para la ingeniería de requisitos de sistemas telemáticos", Revista Colombiana de Computación, vol. 5, jan 2004.
 29. A. Alarcón y E. Sandoval, "Impacto de la ingeniería de requisitos en el desarrollo de proyectos informáticos". Revista Científica, aug 2008.
 30. C. Fraga y J. Reis, "Controla: Herramienta de apoyo al proceso de desarrollo de software en las pequeñas compañías". Revista Ingeniería Informática, apr 2006.
 31. R. Muelas, "Elicitación: cómo conseguir que nos den información". La mente es maravillosa, nov 2017.
 32. D. Cohn, "Análisis de la transparencia en la elicitation de requerimientos al combinar historias de usuario y casos de uso", tesis para optar el grado de magíster en informática con mención en ingeniería de software, Pontificia Universidad Católica del Perú', <http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/7029>, aug 2016.
 33. P. Huertas y E. Vidal, "Una nueva forma de ver la ingeniería de requisitos: + educación de requerimientos", in Memorias del Congreso, 20 Convención Científica de Ingeniería y Arquitectura. V Congreso Internacional de Ingeniería Informática y Sistemas de Información, Universidad Tecnológica de la Habana José Antonio Echevarría, nov 2012.
 34. M. Torrente y E. Piriutella, "Una evaluación a los métodos para eliciar requisitos de seguridad", Ing. USBMed, vol. 2, pp. 48–54, jul 2011. ISSN: 2027-5846.
 35. P. Huertas, "Diseño de guis a partir de casos de uso utilizando criterios de patrones ontológicos de dominio". Libro Electrónico Memorias del XXVI Congreso Nacional y XII

- Congreso Internacional de Informática y Computación XXVI Congreso Nacional y XII Congreso Internacional de Informática y Computación, oct 2013. ISBN 978-607-707-897-5.
- 36. E. Poort and P. With, "Resolving requirement conflicts through non-functional decomposition", Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society, jun 2004.
 - 37. V. Sandana and F. Liu, "Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements", 31st Annual International Computer Software and Applications Conference. IEEE Xplore, aug 2007.
 - 38. H. Abdul, A. Jamil, and U. Imran, "Conflicts identification among non-functional requirements using matrix maps", World Academy of Science, Engineering and Technology, vol. 4, aug 2010.
 - 39. F. Yun, L. Minqiang, and C. Fuzan, "Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix", International Journal of Project Management, vol. 30, p. 363-373, apr 2012.
 - 40. B. Wei, Z. Jin, and D. Zowghi, "An automatic reasoning mechanism for nfr goal models", Fifth IEEE International Conference on Theoretical Aspects of Software Engineering, aug 2011.
 - 41. S. Supakkul, T. Hill, L. Chung, T. Than, and J. Sampaio, "An nfr pattern approach to dealing with nfrs," 18th IEEE International Requirements Engineering Conference. IEEE Xplore, sep 2010.
 - 42. C. Liu, "Cdnfre: Conflict detector in non-functional requirement evolution based on ontologies", Computer Standards & Interfaces. Elsevier, vol. 47, aug 2016.
 - 43. A. Rao and M. Gopichand, "Four layered approach to non-functional requirements analysis", IJCSI International Journal of Computer Science Issues, vol. 8, nov 2011.
 - 44. A. Goknila, I. Kurtev, K. Vandenberg, and W. Spijkerman, "Change impact analysis for requirements: A metamodeling approach", Science Direct. Elsevier, vol. 56, aug 2014.
 - 45. R. Thayer and M. Dorfman, System and software requirements engineering. IEEE Computer Society Press, second ed., 2000.
 - 46. M. Gómez, Notas del Curso Análisis de Requerimientos. Universidad Autónoma Metropolitana. México, first ed., 2011.
 - 47. M. Ventura, La Ingeniería de Requerimientos como factor clave para el éxito de los proyectos de desarrollo de software. Universidad Nacional Autónoma de México, first ed., 2002.
 - 48. I. Sommerville, Ingeniería de Software. Addison - Wesley, seven ed., 2005.
 - 49. M. Jackson, Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. ACM Press, Addison - Wesley, first ed., jun 1995.
 - 50. INTECO, Guía avanzada de gestión de riesgos. Instituto Nacional de Tecnologías de la Comunicación, first ed., dec 2008.

51. PMI, A Guide to the Project Management Body of Knowledge (PMBOK Guide). Project Management Institute, sixth ed., dec 2017.
52. G. Myers, C. Sandler, and T. Badgett, The Art of Software Testing. John Wiley & Sons, Inc., third ed., 2012.
53. S. Zapata, E. Torres, G. Sevilla, L. Aballay, and M. Reus, "Effectiveness of traditional software requirement elicitation techniques applied in distributed software development scenarios". pp. 1-7, XXXVIII Conferencia Latinoamericana en Informática, IEEE Computer Society Press, 2012.
54. T. Nakatani, T. Tsumaki, M. Tsuda, M. Inoki, S. Hori, and K. Katamine, "Requirements maturation analysis by accessibility and stability", pp. 357-364, Software Engineering Conference (APSEC), 2011 18th Asia Pacific, IEEE Computer Society Press, 2011.
55. J. Palmer and R. Evans, "Software risk management: requirementsbased risk metrics". pp. 836-841, Systems, Man, and Cybernetics, Humans, Information and Technology., IEEE Computer Society Press, oct 1994.
56. A. Dutoit and B. Bruegge, "Communication metrics for software development. software engineering", pp. 615-628, IEEE Computer Society Press, 1998.
57. A. Barragans, J. Pazos, A. Fernández, J. García, M. López, R. Díaz, and Y. Blanco, "On the interplay between inconsistency and incompleteness in multi-perspective requirements specifications", pp. 296-321, Information and Software Technology, 2008.
58. A. Zin and N. Pa, "Measuring communication gap in software requirements elicitation process", pp. 66-71, Proceedings of the 8th WSEAS International Conference on Software engineering, parallel and distributed systems, World Scientific and Engineering Academy and Society (WSEAS), 2009.
59. S. Group, "Chaos summary 2010 report", tech. rep., Standish Group, www.standishgroup.com., 2010.
60. S. Ripoll, "Análisis de requisitos de un software de gestión de requisitos". Escuela de Ingeniería. Universidad Autónoma de Barcelona, junio 2014.
61. F. García y P. Puello, "Gestión de requisitos en la ingeniería de software", INGENIATOR - Revista Virtual de los programas de Ingeniería Universidad de San Buenaventura, seccional Cartagena, vol. Vol 1, pp. 57-65, julio 2010. Cartagena de Indias. Colombia.
62. R. Saavedra, L. Ballejos, and M. Ale, "Quality properties evaluation for software requirements specifications: An exploratory analysis", in Memorias del XVI Workshop de Ingeniería en Requisitos WER 2013 (X. I-A. C. on Software Engineering, ed.), CibSE 2013. Universidad de ORT. Uruguay, 2013.
63. F. Levy and P. Muniz, "Analyzing the use of an enterprise model as a stakeholder requirements model: An experiment," in Memorias del XVI Workshop de Ingeniería en Requisitos WER 2013, CibSE 2013. XVI Ibero-American Conference on Software Engineering, 2013.
64. J. Valaski, W. Stancke, S. Reinehr, and A. Malucelli, "Retrospective and trends in requirement engineering through WER", CibSE 2013. XVI Ibero-American Conference on

- Software Engineering, 2013.
- 65. M. Mariduena y A. Estrada, "Priorización de requisitos de software mediante jerarquía de operadores de agregación", XVI Convención y Feria Internacional de Informática, 2016.
 - 66. J. Miguetti y G. Hadad, "Uso de un léxico y escenarios para mitigar amenazas a requisitos en el desarrollo global de software," XIX Ibero-American Conference on Software Engineering, 2016.
 - 67. J. Moreno y M. Marciszack, "Modelando escenarios desde el modelo de negocios empleando patrones", Centro de Investigación, Desarrollo y Transferencia de Sistemas de Información (CIDS), 2017.
 - 68. J. Rojas y D. Carrizo, "Practica industrial en ingeniería de requisitos: Un estudio empírico de empresas latinoamericanas." INCISCOS 2016, 2016.
 - 69. P. Huertas, "Metamodelo de seguimiento para reducir las ambigüedades de los requisitos de software", in Memorias del Congreso., XII Congreso de la Sociedad Peruana de Computación 2013, Universidad Señor de Sipán, Sociedad Peruana de Computación, sep 2013.

Esta obra se terminó de imprimir en el
mes de junio de 2024 en los talleres
gráficos de la Universidad Nacional de
San Agustín de Arequipa



UNSA
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

ISBN 978-61-25136-24-4

9 786125 136244