Universidad de las Ciencias Informáticas
Facultad 4

"Procedimiento para mejorar la Gestión de Requisitos aplicando métricas de calidad en la Unidad de Compatibilización, Integración y Desarrollo".

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autoras: María Caridad Reyes Sondón

Yuricel Sotolongo Montesino

Tutor: Ing.Yordanis Milanés Zamora

Co-tutor: Ing. Violena Hernández Aguilar

Ciudad de la Habana, julio 2008.

"Año del 50 Aniversario del triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser las autoras del presente trabajo de diploma y reconocemos a la Unidad de Compatibilización, Integración y Desarrollo de Productos Informáticos para la Defensa (UCID) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Autores:

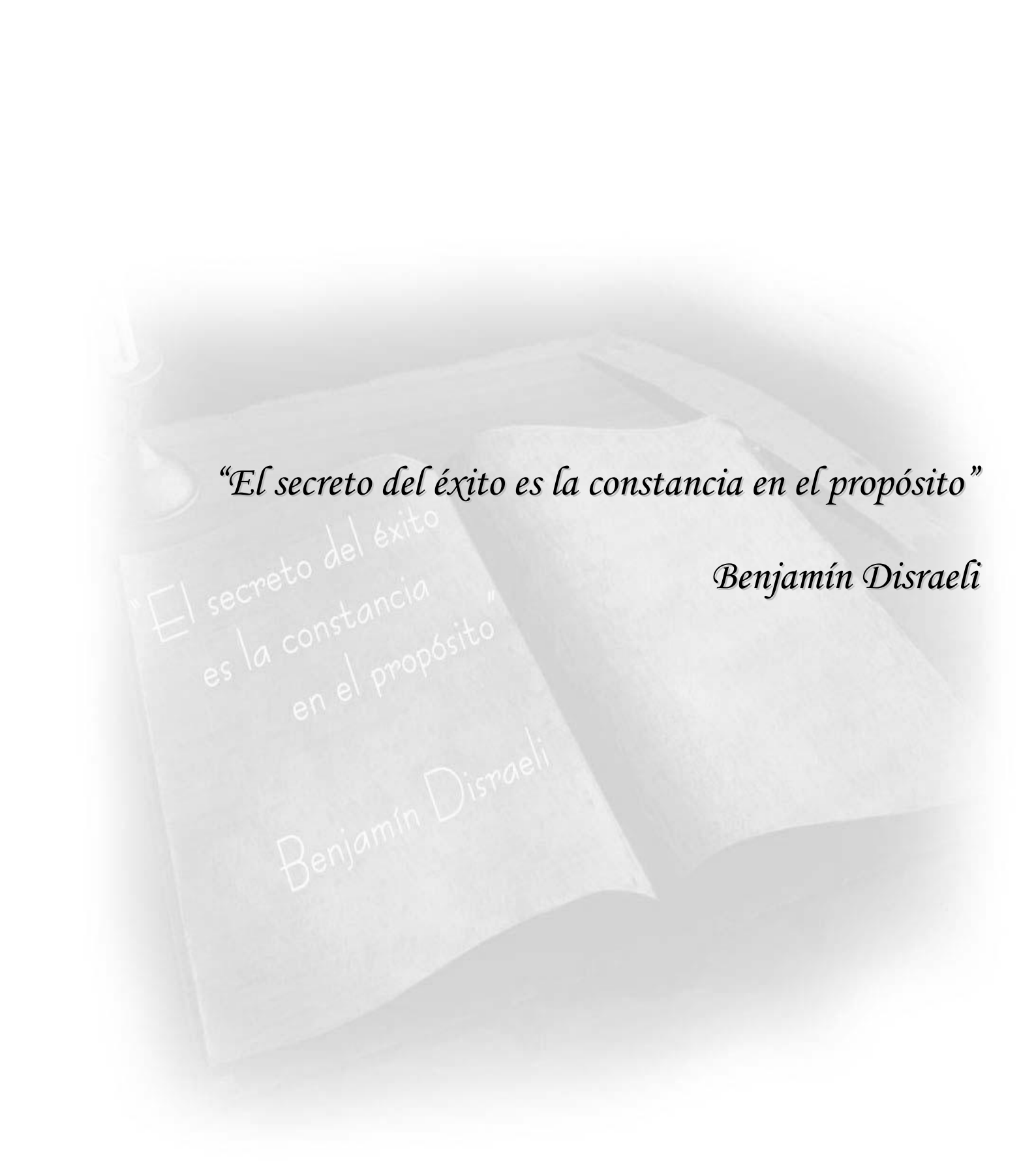
_____ _____

María Caridad Reyes Sondón. Yuricel Sotolongo Montesino.

Tutor: **Co-tutor:**

_____ _____

Ing. Yordanis Milanés Zamora. Ing. Violena Hernández Aguilar.



"El secreto del éxito es la constancia en el propósito"

Benjamín Disraeli

OPINIÓN DEL TUTOR

Título: **"**Procedimiento para mejorar la Gestión de Requisitos aplicando métricas de calidad en la Unidad de Compatibilización, Integración y Desarrollo"

Autores: María Caridad Reyes Sondón
Yuricel Sotolongo Montesino

El tutor del presente Trabajo de Diploma considera que durante su ejecución las estudiantes mostraron las cualidades que a continuación se detallan.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____.

Por todo lo anteriormente expresado considero que las estudiantes están aptas para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de ____.

Ing. Yordanis Milanés Zamora

_____ _____
Firma Fecha

DATOS DE CONTACTO

Tutor: Ing. Yordanis Milanés Zamora
Tel: 837-3135 (residencia UCI) (023) 582375 (Granma) **e-mail:** yordanism@uci.cu

EXPERIENCIA PROFESIONAL

- Jefe de asignatura de Matemática Numérica, Facultad 4, UCI, 2006 – 2007.
- Jefe de asignatura de Sistema Operativo, Universidad de las Ciencias Informáticas, 2007-2008.
- Líder del Módulo Estados Financieros del ERP MINFAR, Enero 2005.
- Asesor de Calidad de Software de la Facultad 4, UCI, Oct. 2007.
- Tutor de trabajos de Diplomas, Facultad 4, UCI, 1er semestre, 2007- 2008.
- Líder del Proyecto Aseguramiento del Polo Aduana, Febrero del 2008.

Cotutora: Ing. Violena Hernández Aguilar
Tel: (07) 2024588 (Ciudad Habana, Playa) **e-mail:** violena@uci.cu

EXPERIENCIA PROFESIONAL

- Ingeniera Informática del Instituto Superior Politécnico José Antonio Echeverría (2005).
- Profesora de la Universidad de las Ciencias Informáticas, en la Disciplina de Ingeniería y Gestión de Software desde el año 2005.
- Cuenta con 3 años de trabajo en la Educación Superior.
- Cursa la Maestría en Gestión de Proyectos Informáticos.
- Se desempeña laboralmente como Especialista General de la Dirección de Calidad de la infraestructura Productiva de la UCI.

AGRADECIMIENTOS GENERALES

A Fidel Castro Rúz por habernos dado la posibilidad de realizar nuestros estudios en esta universidad, y por siempre enseñarnos que: "Toda la gloria del mundo cabe en un grano de maíz".

A la UCI y a la revolución por permitirnos formar parte de este proyecto futuro.

A Violena nuestra asesora, cotutor, consultante y siempre amiga, gracias por sus ánimos y por todo su apoyo incondicional.

A Yudi nuestra amiga de todos los días felices e infelices.

A Yadira, Annelis, Mairelis y Giselle por aguantarnos, por hacernos reír, por cada uno de los momentos en esta universidad.

A nuestras compañeras Nilien, Carmen, Delmys, Maylen y Mercedes por todos estos años de aventura y desventuras.

A nuestros compañeros del grupo de 1er año.

A los profesores que nos han impartido clases en estos 5 años.

A las personas del proyecto calidad y a las que pertenecen al centro UCID por toda su ayuda.

Cary:

A mi mamita por ser lo más bello del mundo, por sus todos sus desvelos, por el amor inmenso, porque es mi vida, mi razón de ser y a ella le debo todo lo que soy y todo lo que seré.

A mi papito por sus enseñanzas, por darme la calma cuando la necesito y la fortaleza siempre.

A mi hermanito, por ser mi alma gemela, por todo su amor y apoyo, por ser mi guía.

A mi cuñada, por soportarme como soy y entenderme, por ser la alegría de mi casa.

A mi sobrina, por tener sus besitos todos los días, por el amor inmenso que me tiene, por ser la niñita de mis ojos.

A mi abuela Vidalina que desde el cielo me acompaña siempre.

A mis padrinos Lily y Miguel por cada apoyo y por toda la fe que me transmitieron en estos años.

A mis tíos (as) July, Rodolfo, Pepe, Gilberto, Ariel, Yola, Chely, Liliana, Leo y María del Carmen por ser mi gran familia.

A mi tía Cecilia por cada correo, por todas las veces que me dijo: "tú si puedes llegar".

A mi tía Vilma por pensar en mí en cada momento, por cada consejo, por quererme tanto.

A mis primos (as) Rafe, Enny, Julito, Alex, Daine y a mi prima Sary por ser fuerte, por llegar hasta donde ha llegado, por ser única.

A Idelisa por ser mi amiga desde pequeña, por no olvidarse de mi a pesar de la distancia y de los años.

A Alionuska mi gran amiga, por sus canciones todos estos años, por ser tan especial.

A Yohandy por ser un amigo sincero, loco, por estar ahí para cualquier duda, por ser mi diccionario.

A Carlos Verdecia, el amigo, el hermano de la universidad, por preocuparse por mí, por quererme tanto como yo a él.

A Yuri por ser mi compañera de tesis más que eso mi amiga, por estar conmigo todos los días, por tener un gran corazón.

A Leo (el novio de Yuri) por el apoyo diario, por escuchar todos nuestros lamentos.

A mis compañeros de aula del 4505 en especial a Yoerkis, gracias por todos los momentos de felicidad.

Yury:

A mis padres por haberme dado la vida, por el amor y dedicación que me han brindado siempre, por estar a mi lado cada vez que los he necesitado, por apoyarme y estar pendiente de mi cada día, por sufrir mi tristeza, por festejar mis triunfos, por todos los sacrificios que han hecho por mí, por la educación que me han dado, por la fuerza que me dan cada día para seguir adelante y ser hoy quien soy, por ser mi razón de luchar cuando creo que no puedo seguir adelante. Por ser en mí vida las personas más importantes.

A mi hermano, por hacerme feliz cada uno de mis días, por hacerme reír cada momento con sus ocurrencias.

A mi prima, por ser como una hermana, por brindarme su ayuda cada vez que necesito de ella, por escucharme y aconsejarme.

A mi abuelo que siempre lo tengo en mi mente y mi corazoncito y aunque hoy no esta con nosotros estaría muy orgulloso de mi en este momento, por haber sido para mi la persona mas especial de este mundo.

A mi abuela, por todo su cariño durante todos estos años, por su preocupación y dedicación.

A Caridad, mi compañera de tesis que ha sido más que una amiga, por su comprensión, ayuda, consejos y apoyo en cada momento. Por ser una gran persona, por tener tan buenos sentimientos, por ser para mi alguien muy importante, por ocupar un lugar especial en mi corazón y por todo lo que hemos vivido durante estos difíciles años.

A Leo por extenderme su mano y estar a mi lado en los momentos mas difíciles y que mas lo he necesitado, por llorar junto conmigo en los momentos de tristeza, por estar siempre pendiente de mi, por comprenderme , soportarme, quererme y hacer cada uno de mis días mas felices.

A Maikel por todo su apoyo y ayuda durante estos cinco años, por todas sus palabras de aliento, por ser para mí como parte de mi familia.

A María por quererme tanto, por ser como mi segunda madre, por darme fuerzas para seguir adelante.

A Yaniris por ser mi amiga de todos los tiempos y más que eso mi hermanita, por escuchar todos mis problemas, por aconsejarme, por estar presente cada vez que la necesito.

A Ailin por los momentos buenos y malos que hemos compartido durante todo este tiempo y por poder contar siempre con ella.

Gracias a todos por estar siempre a mi lado en el transcurso de estos inolvidables años, sin ustedes mi vida no seria tan especial.

RESUMEN

Para obtener un producto de calidad, es necesario que se logren una serie de condiciones en el proceso[1] de desarrollo de software que garanticen el resultado final. El uso de una metodología definida, establecimiento de normas, estándares y métricas para medir la calidad, sin dudas sería un paso de avance en el logro de este propósito.

La Universidad de la Ciencias Informáticas (UCI), desde su creación, ha sido llamada al desarrollo del software[2] como paso importante hacia la instauración de la verdadera Industria Cubana del Software. Encaminada al cumplimiento de este objetivo, se dio a la tarea no solo de producir, sino hacerlo con calidad logrando que el prestigio de la Universidad y del país dependa en gran medida del resultado de sus producciones.

La Gestión de Requisitos (GR) incide de manera importante en la obtención de un mejor producto de software así como en el buen desempeño de los restantes flujos de trabajo por los que este atraviesa. En la Unidad de Compatibilización, Integración y Desarrollo (UCID) perteneciente a la universidad no son desarrolladas las principales actividades de este flujo con la calidad requerida, lo que provoca que no exista control sobre los cambios realizados en los requisitos. La presente investigación aporta un procedimiento de Gestión de Requisitos que permita, mediante el uso de métricas de calidad, guiar a los desarrolladores de software durante el ciclo de vida del producto.

Palabras claves: Gestión de Requisitos, Calidad, Métricas.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el desarrollo de la Industria del Software a nivel mundial la competencia ha aumentado considerablemente y han logrado mantenerse en el mercado aquellas empresas que brinden productos con una mayor calidad, por tal motivo el compromiso en este sentido tiende a elevarse.

La calidad del software es la "concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desee el usuario" [1]. Está dada además por todos los productos y servicios hechos por profesionales preparados y con los procedimientos técnicos correctos que, con costes adecuados, valores, principios éticos y satisfacción de los trabajadores proporcionan éxito a la empresa.

Para obtener una buena calidad de software se necesitan aplicar estándares, mediciones y métricas que nos permitan entender tanto el proceso que se utiliza para desarrollar un producto, como el propio producto. Al investigar y obtener datos sobre un software se pueden calcular y evaluar métricas que conduzcan a incrementar la Calidad en la Gestión de Requisitos de un proyecto.

El producto se mide para lograr un aumento de su calidad, la captura de requisitos[3] es la etapa ideal para definir las métricas a emplear durante su ciclo de vida. Estas establecen la base para lograr un proceso de madurez en la producción de software. Igualmente constituyen estadísticas que se aplican a todos los aspectos de la calidad de software y deben ser medidos desde diferentes puntos de vista: análisis, construcción, funcional, documentación, métodos, procesos y usuarios.

La captura de requisitos no es más que el proceso de averiguar en circunstancias difíciles qué se debe construir, es la etapa donde se sientan las bases para la realización de un sistema con el objetivo de guiar su desarrollo por el camino correcto. El buen desempeño de esta actividad permite realizar menos iteraciones y completar con mayor garantía la primera versión del producto.

Realizando un estudio de las principales causas del éxito o fracaso de los sistemas de software utilizados por 350 empresas entre los años 1997 y 2004, se comprobó que más del 70 % de los proyectos son cancelados o se desvían de sus objetivos. Si se analizan los factores claves de dicho fallo se puede identificar con claridad que los requisitos inconclusos y la poca involucración de los usuarios conducen al fracaso inmediato de los proyectos. De ahí que estos factores sean aspectos a

tener en cuenta durante el proceso de desarrollo del software teniendo como propósito la producción eficiente de un producto que satisfaga las necesidades del cliente. [3]. Figura 1.

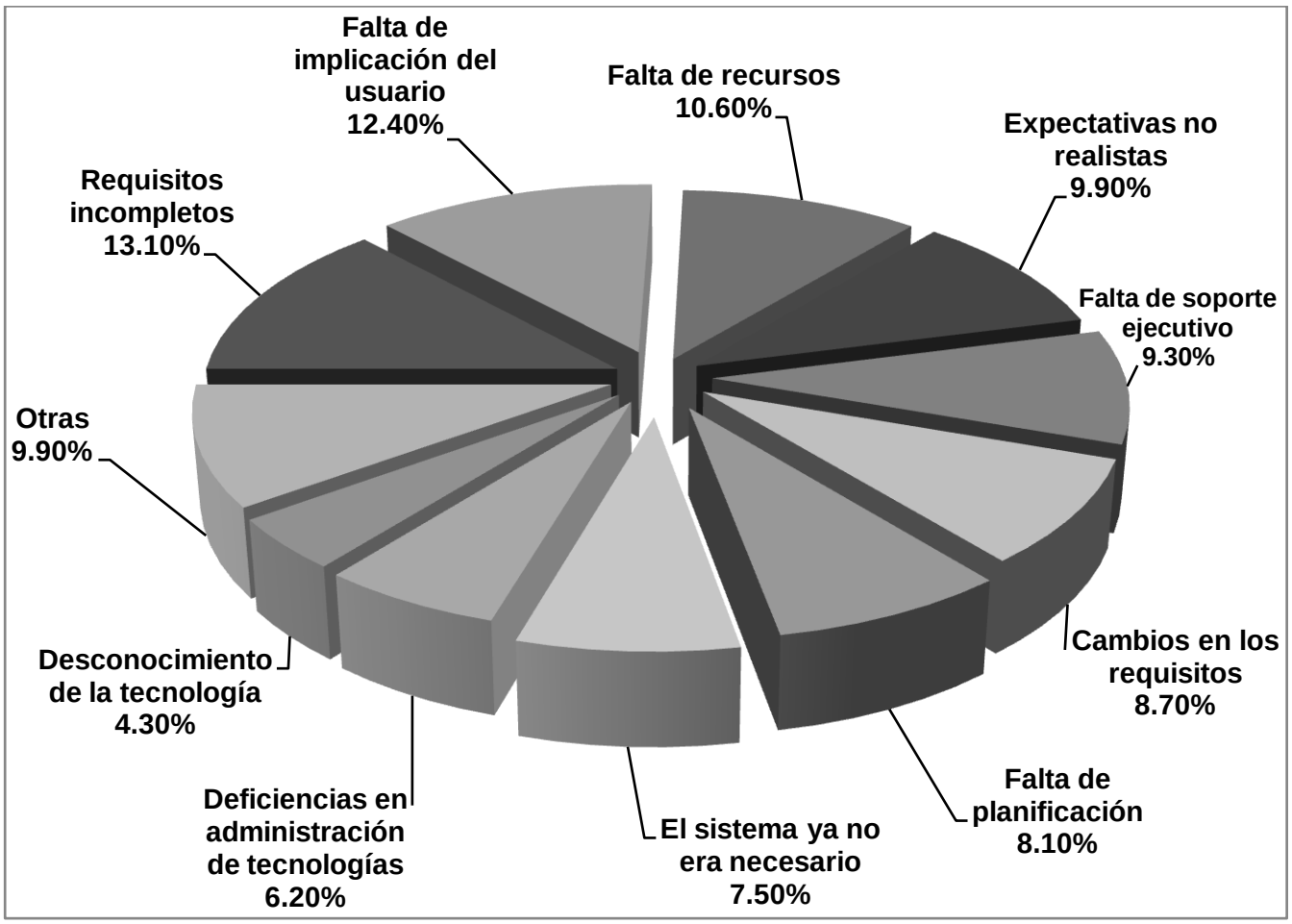


Figura 1: Factores claves en el fallo de la ejecución de proyectos (1997).

Actualmente se conocen métodos, herramientas y procedimientos para llevar a cabo la GR, pero debido a su complejidad este proceso continúa afectando el exitoso desarrollo de los productos software. Para minimizar la complejidad y relatividad inherentes al concepto de calidad en el proceso de GR, se hace necesaria la aplicación de modelos que permitan estimar la calidad del mismo a través de métricas que admitan recopilar datos de forma general.

La Gestión de Requisitos no es más que establecer un entendimiento entre los usuarios y el equipo de desarrollo de software sobre los requerimientos del usuario que serán abordados por el proyecto complementando las métricas para el buen desarrollo del mismo.

Métricas de Calidad: Proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. Es decir cómo medir para que el sistema se adapte a los requisitos que pide el cliente. Son empleadas a nivel mundial por empresas que requieren medir la GR. Las normas más conocidas implantadas por estas métricas son:

- CMM: Modelo de evaluación de los procesos de una organización.
- CMMI: Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software.
- SPICE o ISO /IEC 15504: Evaluaciones de proceso desempeñan un papel importante en la estrategia de organizaciones que desarrollan software o sistemas de software.
- ISO 27001: Publicada en Octubre de 2005. Es la norma principal de requisitos de un Sistema de Gestión de Seguridad de la Información.
- ISO/IEC 12207 Establece un proceso de ciclo de vida para el software que incluye procesos y actividades que se aplican desde la definición de requisitos, pasando por la adquisición y configuración de los servicios del sistema, hasta la finalización de su uso.
- ISO 9001:2000 - Sistemas de Gestión de la Calidad - Requisitos.[2]

El desarrollo de programas de informatización de la sociedad cubana, involucra de manera especial a la Universidad de las Ciencias Informáticas. La producción se concentra en el desarrollo de proyectos en más de 30 Polos Productivos que incluye la relación con entidades nacionales y es una actividad que centra la atención de todos. Dentro del proceso de desarrollo de software que se emprende para la concreción de cada uno de estos proyectos es señalada la captura de requisitos como un pilar importante dirigido a entender las funcionalidades de lo que se va a informatizar y poder enfrentar las fases sucesivas en la elaboración del proyecto de software, logrando que el producto final se obtenga en menor tiempo y con mayor calidad.

Los resultados alcanzados se extienden por los diferentes ministerios del país. El Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR), por ejemplo ha convenido con la UCI en la creación del Centro UCID que acelere el proceso de automatización de este ministerio. A pesar de los esfuerzos realizados se han determinado problemas relacionados con la calidad del producto que se confecciona. Para toda empresa es de vital importancia lograr el aseguramiento de la calidad[4] de sus productos y la UCID no está exenta de ello. La existencia de un Laboratorio de Certificación de Calidad

denota en gran medida el interés por garantizar el crecimiento continuo de una producción de software con calidad en la organización siguiendo las especificaciones de metodologías, estándares y modelos.

Un análisis exhaustivo de los datos puede ayudar a mejorar considerablemente la calidad del proceso. En la UCID no se emplea el procedimiento de la GR como es debido. Los diferentes proyectos trabajan sin llevar a cabo un proceso en el cual se defina la secuencia de cada actividad y que satisfaga las necesidades y expectativas de los interesados, por lo que los resultados no son óptimos. La **situación problémica** que se presenta es que no se documentan, ni se controlan los cambios realizados en la captura de requisitos, lo que dificulta el planteamiento de las métricas aplicables para medir la calidad de este proceso. La no utilización de métricas imposibilita la medición de estos productos en todos los módulos del proyecto y como consecuencia los productos no salen con la mejor calidad.

Se identifica como **problema a resolver** la no realización de actividades fundamentales del procedimiento de GR y la no existencia de métricas que midan la calidad del proceso en la UCID.

Para resolver el problema antes identificado es necesario tomar como **objeto de estudio** el procedimiento para la GR y las métricas para medir la calidad en este proceso, tomando como **campo de acción** la Unidad de Compatibilización, Integración y Desarrollo de software para la defensa.

El **objetivo general** en este trabajo es hacer una mejora del procedimiento de GR y proponer métricas aplicables que permita medir la calidad en este proceso en la UCID.

Para darle cumplimiento al objetivo trazado es preciso definir un procedimiento para capturar los requisitos del software basándose en las propuestas realizadas por varias metodologías. En este caso se utiliza como guía lo expuesto por Rational Unified Process (RUP), metodología definida para el desarrollo de los proyectos de la UCID, pues se adapta con facilidad al ambiente de cambios existente en el MINFAR manteniendo un acuerdo entre los clientes, usuarios finales y equipo de desarrollo en cuanto a lo que debe hacer el sistema. Se definen entonces como **objetivos específicos:**

- Analizar el proceso de GR y las métricas de calidad aplicadas en proyectos de software.
- Diseñar y elaborar métricas de calidad aplicables al proceso de GR.
- Validar la propuesta de mejora del procedimiento para la GR.

Es necesario establecer un conjunto de **acciones de la investigación** a cumplir para lograr alcanzar los objetivos específicos propuestos:

- ♣ Investigar las métricas de calidad para la GR, además del procedimiento de este proceso basándonos en las metodologías existentes.
- ♣ Investigar el estado actual de las métricas de calidad para la GR en la UCI.
- ♣ Investigar como se realiza el procedimiento de la GR en la UCID.
- ♣ Determinar que métricas de calidad para la GR son aplicables a la UCID.
- ♣ Identificar posibles beneficios de la aplicación de métricas de calidad para la GR en la UCID.

Posibles resultados a obtener:

- ♣ Obtener un procedimiento con la aplicación de métricas para llevar a cabo la GR que permita medir la calidad de este proceso en la UCID.
- ♣ Definir las actividades necesarias para desarrollar la GR en los proyectos productivos asociados a la UCID.

Entre los **Métodos Científicos** que mayormente guían el desarrollo de esta investigación se emplearon combinaciones de los métodos teóricos y métodos empíricos**:**

Métodos teóricos:

- ♣ **Analítico – sintético.** Al buscar la esencia del problema de investigación y los rasgos que lo caracteriza y distingue; extrayendo los elementos más importantes que se relacionan con el objeto de investigación.
- ♣ **Histórico – lógico.** Al hacer un estudio crítico del los trabajos anteriores en este contexto y para utilizarlos como punto de referencia y comparación de los resultados alcanzados, en toda su trayectoria.
- ♣ **Modelación.** Al descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio, explicando las razones por las cuales el procedimiento propuesto es el que más se ajusta a las características del proceso de captura de requisitos.

Métodos empíricos:

- ♣ Entrevista.
- ♣ Encuesta.

La tesis se encuentra estructurada en **Resumen, Introducción,** tres **Capítulos, Conclusiones, Recomendaciones, Referencias Bibliográficas, Bibliografías, Anexos y Glosario**.

En el **Capítulo I** se realiza un análisis profundo sobre la calidad, la Gestión de Requisitos y las métricas aplicadas al software. Se explican como se lleva a cabo la GR en algunas metodologías analizando el aporte de estas a RUP. Se hace un estudio sobre la captura de requisitos y los cambios en ellos tomando como inicio su estado actual en el Centro UCID.

En el **Capítulo II** se explica como proceder al desarrollo de los pasos principales del procedimiento que se propone, dejando identificado los artefactos y roles de cada actividad. Además se establecen un grupo de métricas de calidad, técnicas y plantillas para la mejora de este procedimiento.

En el **Capítulo III** se exponen las características, la disponibilidad y el grado de conocimiento de los sistemas de expertos, realizando la validación de la propuesta con la aplicación del Método Delphi. Para la aplicación del método se tuvo en cuenta la participación de un grupo de expertos de la Universidad.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

Una buena captura de requisitos incide en la calidad de los productos entregados. Este capítulo servirá para fundamentar el estado del arte a través de la revisión bibliográfica realizada, investigar y argumentar, sobre la Gestión de Requisitos en varias metodologías así como las métricas de calidad del software aplicables a este proceso que posibilitan evaluarlo. Se abarcan los conceptos fundamentales a tener en cuenta, empezando por la calidad, como principal objetivo a la hora de desarrollar un producto de software; la GR de forma general, analizando cuidadosamente los problemas que se pueden encontrar en este proceso, de igual modo son analizadas las métricas de calidad como proveedoras de la información necesaria para la toma de decisiones técnicas.

1.2. Calidad del Software.

1.2.1 Definición de Calidad.

La calidad de un producto del software debe evaluarse usando un modelo de calidad que tiene en cuenta criterios para satisfacer las necesidades[5] de los desarrolladores, mantenedores y usuarios finales.

"La calidad es un conjunto de características de una entidad[6] que le confieren la aptitud para satisfacer las necesidades establecidas y las implícitas."[4]

También existe otra definición referente a la calidad: "La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario."[5]

Los modelos de calidad[7] pueden ser utilizados para construir mejores productos y asegurar su calidad. Construir un modelo de calidad es bastante complejo y es usual que estos modelos descompongan la calidad del software jerárquicamente en una serie de características y sub-características que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad.

Modelo de Calidad: Es un conjunto de buenas prácticas para el ciclo de vida del software enfocado en procesos de gestión y desarrollo de proyecto.

Calidad del Software:

- ♣ Los requisitos del Software son la base de las medidas de calidad.

- Unos estándares específicos definen un conjunto de criterios de desarrollo que guían la manera en que se hace la ingeniería del Software.
- Existe un conjunto de requisitos implícitos que ha menudo no se nombran. Si el software cumple con sus requisitos explícitos pero falla en los implícitos, la calidad del software no será fiable.

1.2.2 Aseguramiento de la calidad. Actividades de SQA.

La garantía de calidad del software comprende una zona de gran variedad de tareas asociadas con dos constitutivos diferentes, los ingenieros de software que realizan trabajo técnico y un grupo de Aseguramiento de la Calidad del Software (SQA) que tiene la responsabilidad de la planificación de garantía de calidad, supervisión, mantenimiento de registro, análisis e informes.

Los ingenieros de software afrontan la calidad (y realizan garantía de calidad) aplicando métodos técnicos, sólidos y medidas, realizando revisiones técnicas formales llevando a cabo pruebas de software bien planificadas. Las reglas de grupo de SQA tratan de ayudar al equipo de ingeniería de software en la consecución de un producto final de alta calidad.

Estas son las actividades que realizan (o facilita) un grupo, independiente de SQA:

1. Establecimiento de un plan de SQA para un proyecto.
2. Participación en el desarrollo de la descripción del proceso de software.
3. Revisión de las actividades de ingeniería de software para verificar su ajuste al proceso del software definido.
4. Auditoria de los productos de software designados para verificar el ajuste con los definidos como parte del proceso de software.
5. Asegurar que las desviaciones del trabajo y los productos del software se documenten y se manejen de acuerdo con un procedimiento establecido.
6. Registrar lo que no se ajuste a los requisitos e informarlos a los superiores.

1.2.3 Modelos y estándares de calidad.

El modelo CMMI (Capability Madurity Model Integration) se presenta como un modelo de mejora de procesos en desarrollo de software que puede complementarse a la norma ISO9001. Sin embargo uno de los grandes desafíos es llegar a encontrar una estrategia que permita realizar este tipo de integración de forma “natural”.

El estándar de calidad ISO 9001

ISO 9001 es el estándar más importante internacionalmente. El estándar, que ha sido adoptado por mas de 130 países para su uso, se está convirtiendo en el medio principal con el que los clientes pueden juzgar la competencia de un desarrollador de software. Uno de los problemas con el estándar ISO 9001 está en que no es específico de la industria: está expresado en términos generales, y puede ser interpretado por los desarrolladores de diversos productos. Se han realizado muchos documentos que relacionan el estándar con la industria del software, pero no entran en una gran cantidad de detalles.

ISO9001 como norma de gestión de la calidad, exige definir una serie de procesos de la organización que permitan conseguir los objetivos de calidad y tiende el puente entre los requerimientos del cliente y la satisfacción del mismo, definiendo para ello procesos estratégicos, claves y de soporte. ISO9001 es suficiente para llegar a institucionalizar la mejora continua de los procesos, sin embargo, contar con un modelo o guía de buenas prácticas en desarrollo de software, como lo es CMMI, ayuda a mejorar los procesos de producción de software.

Enfoque en los procesos de CMMI

ISO9001 hace mucho énfasis en la definición, control y mejora de los procesos que permitan garantizar la calidad de los productos y servicios producidos y que se encuentran entre los requerimientos y la satisfacción del cliente. Los procesos se han agrupado en tres grandes categorías, los relativos al soporte de la gestión, los procesos clave que hacen operativa la gestión y los estratégicos que permiten llegar a los objetivos planteados. En la trayectoria de implementación de un modelo CMMI, los primeros pasos están centrados en la definición de los procesos, antes que en el producto o la tecnología.[6]

La definición del proceso, ha dividido las siguientes etapas principales:

- Determinación: La determinación y definición es la actividad mediante la cual se interpreta, analiza, modela y valida la información que determinan tareas principales en el proceso.
- Documentación: Actividad en la cual se realiza el respectivo detalle de los documentos (contenido y formatos) generados en el proceso.
- Análisis y Negociación: Permite alcanzar un acuerdo y/o resolver problemas debido a los cambios que se pueden producir en el proceso. Como cada usuario, cliente o miembro del equipo tiene su propia apreciación, se debe tratar de cubrir y entender tales apreciaciones.

- ♣ Verificación: Se utilizan métodos que permitan realizar el control y validación del proceso, en el modelo CMMI.

Para la definición de los procesos se ha realizado un análisis y compendio de varios estándares, especialmente los de la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) relativos al desarrollo de software. Se toma siempre en cuenta la especificación CMMI, previa a la definición del proceso y en cada etapa se vincula con el objetivo general o la práctica específica del modelo CMMI. La tabla 1 muestra una vinculación de las etapas principales del proceso y las prácticas específicas para la GR.

Gestión de Requisitos	CMMI Prácticas Especificas
Definición de Requerimientos	SP1.1 Obtención y entendimiento de los requerimientos.
Análisis & Negociación	SP1.2 Obtener el acuerdo sobre los requerimientos (criterios de aceptación).
Documentación	SP1.1 Obtención y entendimiento de los requerimientos.
Validación	SP1.2 Obtener el acuerdo sobre los requerimientos. SP1.5 Identificar inconsistencias entre los productos de trabajo del proyecto y los requerimientos.
Gestión de los cambios	SP1.3 Gestión de los cambios de requerimientos. SP1.4 Mantener la trazabilidad de los requerimientos. SP1.5 Identificar inconsistencias entre los productos de trabajo del proyecto y los requerimientos.

Tabla 1: Vinculación de las etapas del proceso con las prácticas específicas de CMMI.

Bajo la especificación de las tareas de ISO 9001 para cada una de las etapas del proceso de desarrollo de software, que se ha definido, se toma en consideración como práctica para la realización de la tarea, las prácticas específicas (SP) del modelo CMMI como se muestra en la tabla 2.

Tarea (Espec. ISO 9001)	Producto	Prácticas CMMI
ASI 2	Establecimiento de Requisitos	
ASI 2.1	Obtención de requisitos	SP1.1 Obtención y entendimiento de los requerimientos.
ASI 2.2	Especificación de casos de uso	

ASI 2.3	Análisis de Requisitos	SP1.1 Obtención y entendimiento de los requerimientos. SP1.2 Obtener el acuerdo sobre los requerimientos. SP1.4 Mantener la trazabilidad de los requerimientos.
ASI 2.4	Validación de Requisitos	SP1.5 Identificar inconsistencias entre los productos de trabajo del proyecto y los requerimientos.

Tabla 2: Vínculo de la especificación de tareas ISO y CMMI.

1.3. Gestión de Requisitos en Metodologías.

1.3.1 Gestión de Requisitos en eXtreme Programming (XP).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa fundamentalmente en la realimentación continua entre el cliente y el equipo de trabajo, la comunicación entre todos los participantes, y el enfrentamiento de los cambios. Es usado en proyectos pequeños con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico.

Desde el punto de vista que describe Pressman, el proceso de Ingeniería de Requisitos (IR) que refiere la metodología XP va dejando atrás lo fundamental de la investigación, que es la GR.

Uno de los procesos que define es la identificación de los requisitos con los que contará el sistema, especificando claramente los objetivos del sistema o producto, e investigando cómo se ajustan a las necesidades del negocio. Los requisitos son recopilados en reuniones en las que participan clientes e ingenieros del sistema. Hay que añadir que ese es el momento de solucionar cualquier problema de alcance, o comprensión.

El análisis y la negociación de requisitos, en XP se tratan a través de las historias de usuarios. El cliente describe brevemente las características que el sistema debe poseer como parte de la

identificación de requisitos, en muchos casos sólo se propone utilizar un nombre y una descripción, mas quizás una estimación de esfuerzo en días.

Se opina que la sola descripción de una historia de usuario no es suficiente, pero siguiendo la sencillez de XP, cualquier otra información asociada podría estar justificada con respecto al beneficio que aporte. Es decir, la información de una historia de usuario podría variar y ajustarse a las características específicas del proyecto.

En XP no se define la especificación de requisitos, pues se sugiere realizar en grandes sistemas, y es aplicada generalmente en proyectos pequeños. Tampoco crea un modelado del sistema, pues según la literatura estudiada no se refleja que evalúe los componentes del sistema y sus relaciones entre sí, ni determina cómo están reflejados los requisitos, ni cómo se ha concebido la estética en el sistema.

Todo proyecto sea grande o pequeño debe abordar la validación de requisitos[8], para asegurar que todos han sido establecidos sin ambigüedad, inconsistencias y se ajusten a los estándares establecidos en un comienzo para el proceso, proyecto y producto. XP realiza semanalmente reuniones entre los clientes y desarrolladores con la finalidad de analizar, el avance en el proyecto, así como las técnicas y estrategias que se deben seguir.

Respecto a la GR expuesta, no concuerda con lo planteado. Una vez identificados los requisitos, en esta metodología no se les asigna identificadores, ni se crean matrices para su seguimiento. Este proceso se determina mediante una guía [7]. Su principal objetivo es orientar al usuario a la hora de concebir o seguir un cambio en una historia.

En fin XP no realiza las actividades propuestas por la GR que plantea la identificación de cada requisito y la creación de las dependencias entre los mismos mediante el uso de las matrices y la trazabilidad. No refiere estas actividades antes de realizar un cambio en los requisitos, o sea que las realiza sin su previa definición.

1.3.2 Gestión de Requisitos en SCRUM.

SCRUM es una metodología ágil basada en el desarrollo incremental e iterativo. Se basa fundamentalmente en la gestión de proyectos de software. No es una metodología de análisis, ni de diseño, como podría ser RUP. Sólo abarca prácticas de gestión sin entrar en las prácticas de desarrollo como lo hace XP. Es empleada en entornos que trabajan con requisitos inestables y que

requieren rapidez y flexibilidad; donde se presenten situaciones frecuentes en el desarrollo de determinados sistemas de software.

Para controlar los requisitos define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP que, centrados en prácticas de tipo técnico, carecen de ellas. Entre sus elementos básicos se encuentran Product Backlog y Sprint Backlog, los que se explicarán más adelante.

SCRUM al igual que XP realiza la identificación de los requisitos, organizándolos en una lista denominada Product Backlog, quedando determinados los objetivos y el alcance del sistema. Los requisitos son recopilados mediante reuniones con el cliente, y no es necesario que desde un principio queden todos definidos, sino que se pueden ir incorporando a la lista a medida que el proyecto se desarrolle.

Una vez coleccionados los requisitos se procede a su análisis y negociación. Son ordenados de menor a mayor en otra lista llamada Product Backlog en dependencia de la prioridad que se les confiera. Cualquier persona tiene privilegios para añadir funcionalidades al Product Backlog, pero sólo una puede ordenarlo. A esta persona se le denomina Product Owner y es el responsable del producto final. Posteriormente del Product Backlog, se toman las primeras funcionalidades, y son descompuestas en pequeñas tareas las cuales son anotadas en otra lista denominada Sprint Backlog, y constituyen las tareas próximas a implementar.

Al final del período establecido para implementar los requisitos que se seleccionaron en la lista Sprint Backlog, se presenta el producto y se toma del Product Backlog ordenado las funcionalidades que serán implementadas próximamente.

Según lo establecido al inicio como GR, SCRUM no maneja este término completamente, se limita solo a la identificación, análisis y validación de los requisitos sin determinar su seguimiento y control, es decir solo pone en práctica alguna de las actividades que describe la propia gestión.

1.3.3 Gestión de Requisitos en Microsoft Solution Framework (MSF).

Una de las metodologías tradicionales de desarrollo es MSF, compleja, adaptable, flexible y con tecnología agnóstica porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. Entres sus ventajas principales se encuentra la posibilidad de adaptarse a proyectos de

cualquier tamaño y complejidad. La base de dicho modelo comprende prácticas y técnicas de otras metodologías. Permite optimizar recursos, manejar riesgos y administrar cambios para un óptimo beneficio a los usuarios. Es la metodología empleada por Microsoft para el desarrollo de software.

En una de sus fases de desarrollo, denominada visión y alcance identifica las metas y objetivos a alcanzar, término que le atribuye a requisito de software. Deben ser determinadas por los líderes y responsables del proyecto y respetarse durante la ejecución del proyecto en su totalidad.

El equipo debe tener una visión clara de lo que se quisiera lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo. Además utiliza la lista de fuentes de requisitos para hacer su recopilación. Las metas son descritas en el documento de visión/ámbito

Posteriormente en la fase de planificación se procede a realizar otro de los procesos definidos: el análisis y negociación de los requisitos, que en MSF se le atribuye el nombre de descomposición del trabajo.

El equipo de trabajo prepara las especificaciones funcionales, o metas dividiéndolas en partes más pequeñas y manejables, y asignándole una prioridad. Aquellas funcionalidades de mayor prioridad serán implementadas en la fase más temprana del proyecto, ampliándose al máximo el tiempo disponible para reaccionar ante los problemas.

En el documento visión/ámbito, cuando se establece su línea base, ya el equipo considera que todas las funcionalidades están bajo control. Por lo que los cambios en el ámbito deben ser revisados y aprobados tanto por el equipo como por el cliente.
Parte de un buen control del cambio implica tomar buenas decisiones sobre las compensaciones. Los cambios son controlados mediante la matriz de compensaciones, o el triángulo de tres variables interdependientes. MSF utiliza herramientas para dicho control.

1.3.4 Gestión de Requisitos en Business Process Management (BPM).

Business Process Management (BPM) es un conjunto de herramientas tecnológicas que junto con una nueva filosofía de negocio permite diseñar la arquitectura empresarial modelando los procesos de negocio mediante flujos de trabajos, automatizando su funcionamiento de principio a fin y permitiendo su monitorización y control.

El BPM es una herramienta útil para hacer una toma de requisitos en un proyecto de reingeniería o mejora de un proceso de una empresa. Para mejorar dicho proceso podemos hacerlo de varias formas, automatizándolo, es decir, no variar el proceso y automatizar las funciones que lo forman; o cambiando el proceso, de forma radical (reingeniería) o sólo realizando mejoras puntuales. Para cada tipo de mejora la toma de requisitos es distinta, ya que no es lo mismo si el objetivo es desarrollar software, integrar un software estándar, o integrar aplicaciones heredadas tanto para automatizar el proceso completo como tramos del mismo. Sea cual sea el objetivo, el modelado del proceso es fundamental, sirve para conocerlo, analizarlo y poder así proceder a su mejora.

Si el objetivo es desarrollar un Sistema de Información (SI) personalizado es necesario añadir a la toma de requisitos el diseño mediante diagramas UML (Unified Modeling Language), que no sólo documentan el sistema, sino que herramientas como Rational transforman directamente en código.

1.3.5 Gestión de Requisitos en Rational Unified Process (RUP).

RUP es un proceso de Ingeniería del Software que proporciona una visión disciplinada para la asignación de tareas y responsabilidades en las organizaciones de desarrollo de software. Integra un conjunto de “buenas prácticas“ para el desarrollo de software en un marco de procesos válido para un rango amplio de tipos de proyectos y organizaciones, dentro de estas se encuentra la GR, aspecto tratado en este trabajo.

La ingeniería de requisitos en RUP comienza con su identificación. Desde el comienzo del proyecto en la fase de inicio, los requisitos son recogidos mediante entrevistas con el cliente y son guardados en el documento Visión. En este documento también se recogen los cambios que se puedan introducir. Posteriormente se procede al análisis y negociación de los mismos. Los requisitos pueden ser definidos en casos de uso, casos de prueba, características funcionales y no funcionales.

La validación de los requisitos en esta metodología se realiza a través de los casos de prueba que son definidos para cada uno de los requisitos funcionales del sistema. En ellos se determina si han cumplido o no con su funcionalidad.

Reemplazar un requisito no solo se reduce a quitar un requisito y sustituirlo por uno nuevo, sino también hay que analizar el impacto que este cambio puede ocasionar en los ya existentes. Es necesario entonces cerciorarse de dar a los requisitos una estructura que sea resistente a los cambios. Como solución, RUP utiliza la trazabilidad para representar las dependencias entre los

requisitos y otros artefactos del ciclo de vida del desarrollo y de esa manera gestionarlos. Incluso, hace énfasis en que si un requerimiento no ha sido cambiado, pueden modificarse los atributos y la trazabilidad asociados a ellos. Esto puede o no ocurrir, el encargado de mantener esta información sobre una base en curso es el analista del sistema que cuenta con una solución ante el impacto que puede ocasionar un cambio en los requisitos. El manejo del cambio incluye actividades como establecer una línea base, determinándose las dependencias que son importantes resaltar.

La metodología RUP es la que más se acerca al concepto de IR y dentro de este al de gestión que se define en “Ingeniería del Software. Un enfoque practico“. En él la gestión, como se ha analizado, se realiza a través de la trazabilidad entre los diferentes elementos, lo que permite el seguimiento de los cambios en los mismos a lo largo del ciclo de vida del proyecto.

1.4. Gestión de Requisitos.

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refina en detalle los requisitos del sistema y el papel asignado por el ingeniero del sistema. Se crea modelos de los requisitos de datos, flujos de información y control, y del comportamiento operativo. Se analizan soluciones alternativas y el modelo completo del análisis es creado.

Se describe el proceso de IR del software como: “La ingeniería de requisitos es el uso sistemático de procedimiento, técnicas, lenguajes, herramientas para obtener con un coste reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario” [8].

1.4.1. Captura de requisitos: de la visión a los requisitos.

1. ¿Por qué la captura de requisitos es complicada?

Captura de requisitos: es el proceso de averiguar en circunstancias difíciles qué se debe construir. Los desarrolladores no pueden escribir un código sin saber qué es lo que debe hacer, algo que sucede en algunas ocasiones. Los analistas documentaban requisitos según lo que los usuarios pedían, pero llegaba a cientos de páginas y no podían concretarse fácilmente. Los usuarios sabían bien qué debía hacer el software recién cuando el producto estaba casi terminado y para hacer los cambios pedidos no quedaba otra que postergar las fechas y aumentar presupuesto, es decir el usuario no sabía cuáles eran los requisitos.

2. Objetivo de flujo del trabajo de los requisitos.

Objetivo: Guiar el desarrollo hacia el sistema correcto.

Esto se consigue mediante la descripción de los requisitos del sistema (es decir, condiciones o capacidades que el sistema debe cumplir) suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente (incluyendo al usuario) y los desarrolladores sobre que debe y que no debe hacer el sistema. El resultado del flujo de trabajo de los requisitos también ayuda al jefe de proyecto a planificar las iteraciones y versiones del cliente.

3. Visión general de la captura de requisitos.

Para la captura de requisitos ciertos pasos son factibles en la mayoría de los casos, los que nos permite sugerir un flujo de trabajo arquetípico. Este flujo incluye los siguientes pasos, que se llevan a cabo conjuntamente:

Enumerar los requisitos candidatos: De aquí se obtienen características: lista de sugerencias que el usuario va dando. Aumenta cuando se agregan elementos; se restan al convertirse en otros artefactos como casos de uso. Compuesto por un nombre corto, breve descripción y un conjunto de valores:

- Estado (propuesto, aprobado, validado) Coste estimado, Prioridad, Nivel de Riesgo.
- Estos valores sirven para calcular tiempo que llevará el proyecto y cómo dividirlo en iteraciones.

Comprender contexto del sistema: Hay 2 aproximaciones para expresar el contexto de sistema.

- Modelo de dominio: Describe los objetos del dominio, se les asignan un nombre que se pasan a un glosario para mejorar la comunicación entre las personas que trabaja. Los objetos ayudan a identificar clases.
- Modelo de negocio: Es más amplio que el modelo de dominio. Describe los procesos que componen el negocio. Su objetivo es comprender cuáles son los procesos que soportará el sistema.

Capturar requisitos funcionales: Se basa en los caso de usos. Describen de qué forma el usuario va a utilizar el sistema. Cada usuario requiere de varios casos de uso. Los analistas proponen cómo será la interfaz del sistema esbozando varias versiones para que el usuario decida.

Capturar requisitos No funcionales: Especifica las propiedades del sistema que tienen que ver con rendimiento, velocidad, uso de memoria, plataforma. La fiabilidad es el tiempo de respuesta media, defectos[9] por miles de líneas de código. Imponen condiciones a los requisitos funcionales.

Puede que no pertenezca a ningún caso de uso => se agregan como requisitos adicionales.

4. Papel de los requisitos en el ciclo de vida de software
   - ♣ Inicio: Se identifican la mayoría de los casos de uso para detallar los más importantes (10%).
   - ♣ Elaboración: Se captura los requisitos para estimar tiempo de proyecto (80%).
   - ♣ Construcción: Se capturan e implementan los demás requisitos.
   - ♣ Transición: No hay captura de requisitos.

1.4.2. Optimización del proceso de Gestión de Requisitos en el desarrollo de aplicaciones del software.

Una de las especificaciones informáticas más antiguas y más importantes para el buen funcionamiento de una infraestructura informática de cualquier empresa son los requisitos. Éstos permitirán la resolución de problemas y lograrán un mayor impacto en la productividad de los sistemas.

Una eficiente GR a lo largo de todo el ciclo de vida del proyecto contribuye eficazmente a la calidad del producto final y al grado de satisfacción del cliente.

Los requisitos son definidos durante las fases más tempranas del desarrollo de sistemas informáticos, y pueden verse como la especificación de lo que debería ser implementado. Por lo tanto, y de forma más concreta, la IEEE [9] los define como:

- ♣ Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
- ♣ Una condición o capacidad que debe estar presente en un sistema o componentes de sistema. para satisfacer un contrato, estándar, especificación u otro documento formal.
- ♣ Una representación documentada de una condición o capacidad.

Estos requisitos han de ser vistos desde un punto de vista muy relacionado con la ingeniería, lo que implica el uso de técnicas sistemáticas y repetibles para asegurar que los requisitos del sistema son completos, consistentes y relevantes [10]. Sin estas técnicas, el equipo de desarrollo:

- ♣ No sabe cuales son las metas a lograr.
- ♣ No pueden inspeccionar y probar su trabajo de manera adecuada.
- ♣ No se puede controlar su productividad.
- ♣ No obtiene datos adecuados de sus prácticas.
- ♣ No puede predecir el tamaño y esfuerzo del siguiente proyecto.
- ♣ No puede satisfacer a sus clientes.

No hay ingeniería profesional sin requisitos bien gestionados. Todas estas actividades relacionadas con la ingeniería de requisitos, hasta hace no mucho tiempo, eran acometidas de manera absolutamente manual, incluyendo el uso de procesadores de texto para acometer su gestión.

En la actualidad persisten problemas en el desarrollo de software, entre ellos, un inadecuado entendimiento de las necesidades de los usuarios, incapacidad de absorber cambios en los requisitos e insatisfacciones de los clientes por inaceptable o bajo desempeño del software. Las principales causas son la administración insuficiente de requisitos; los problemas que afectan la comunicación; las inconsistencias no detectadas entre requisitos, diseño y programación; las validaciones tardías de requisitos; el enfrentamiento reactivo de riesgos y la propagación de cambios sin control.

Los modelos de proceso de IR, a pesar de su evolución, aún presentan carencias. Por tanto, para obtener un producto de calidad, se requiere una mejora en los procesos de IR. El objetivo es proporcionar un procedimiento para efectuar la GR de un proyecto de software basado en la integración de las mejores prácticas, con un enfoque estratégico que potencie de manera efectiva el desempeño del proceso de gestión de desarrollo del proyecto de software y la satisfacción del cliente.

1.4.3. Identificación de requisitos para el software.

Antes que los requisitos puedan ser analizados, modelados o especificados, deben ser recogidos a través de un proceso de obtención. Un cliente tiene un problema que pretende sea resuelto con una solución basada en computadora. Un desarrollador responde a la solicitud de ayuda del cliente. La comunicación ha empezado. Pero el camino de la comunicación al entendimiento está a menudo lleno de baches.

Inicio del proceso

La técnica de obtención de requisitos mas usada es llevar a cabo una reunión o entrevista preliminar. Gause y Weinberg [11] sugiere que el analista empiece preguntando cuestiones de contexto libre. Es decir, un conjunto de preguntas, que llevarán a un entendimiento básico del problema, que solución busca, la naturaleza de la solución que se desea y la efectividad del primer encuentro. El primer conjunto de cuestiones de contesto libre se enfoca sobre el cliente, los objetivos generales y los beneficios esperados. Por ejemplo, el analista podría preguntar:

- ¿Quién está detrás de la solicitud de este trabajo?
- ¿Quién utilizará la solución?
- ¿Cuál será el beneficio económico del existo de una solución?

- ¿Hay alguna otra alternativa para la solución que necesita?

Estas preguntas ayudan a identificar todas las participantes que tienen un interés en el software a construir. Además las preguntas identifican los beneficios medibles en una implementación correcta de posibles alternativas para un desarrollo normal del software.

El siguiente conjunto de preguntas permite al analista obtener un mejor entendimiento del problema y al cliente comentar sus opiniones sobre la solución:

- ¿Cómo caracterizar una buena salida (resultado) generada para una buena solución?
- ¿A qué tipo de problema va dirigida esta solución?
- ¿Puede mostrarme o describirme el entorno en que se utilizará la solución?
- ¿Hay aspecto y restricciones especiales del rendimiento que afecten a la manera de enfocar la solución?

El último conjunto de preguntas se concentra en la eficacia de la reunión Gause y Weinberg, proponen la siguiente lista:

- ¿Es usted la persona adecuada para responder estas preguntas? ¿Sus respuestas son oficiales?
- ¿Estoy preguntando demasiado?
- ¿Hay alguien más que pueda proporcionar información adicional?
- ¿Hay algo más que debería preguntarle?

Estas preguntas y otras ayudarán a romper el hielo e iniciar la comunicación tan esencial para el éxito del análisis. Pero el formato de reunión tipo pregunta y respuesta no es un enfoque que haya tenido mucho éxito. De hecho esta sesión de preguntas y respuesta deberían emplearse solamente en el primer encuentro y sustituirse después por una modalidad que convine elemento de resolución del problema, negociación y especificación.

Técnicas para facilitar las especificaciones de una aplicación

Los clientes y los ingenieros del software a menudo tienen una mentalidad inconsciente de nosotros y ellos. En vez de trabajar como un equipo para identificar y refinar los requisitos, cada uno define por derecho su propio territorio y se comunica a través de una serie de memorandos, papeles de posiciones formales, documentos y sesiones de preguntas y respuestas. La historia ha demostrado

que este método no funciona muy bien. Abundan los malentendidos, se omite información importante y nunca se establece una buena relación de trabajo [12].

Directrices básicas para la utilización de TFEA (Técnicas para Facilitar las Especificaciones de una Aplicación):

- La reunión se celebra en un lugar neutral y acuden tantos los clientes como los desarrolladores.
- Se sugiere una agenda lo suficientemente formal como para cubrir todos los puntos importantes, pero lo suficientemente informal para animar el libre flujo de ideas.
- Un coordinador que puede ser un cliente, un desarrollador o un tercero que controla la reunión.
- Se usa un mecanismo de definición (que pueden ser hojas de trabajo, gráficos o carteles).
- El objetivo es identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos de la solución en una atmósfera que permita alcanzar el objetivo.

Una vez terminadas las reuniones para las TFEA cada asistente hace una lista de criterios de validación del producto-sistema y presenta su lista al equipo. Se crea así una lista de consenso de criterios de validación. Finalmente, uno o más participantes o algún tercero es asignado para escribir el borrador entero de especificación con todo lo expuesto en la reunión TFEA.

TFEA no es la panacea para los problemas que se encuentran en las primeras reuniones de requisitos pero el enfoque de equipo proporciona las ventajas de muchos puntos de vistas, estudio y refinamiento instantáneo y un paso de avance hacia el desarrollo de una especificación.

Despliegue de la función de calidad

El despliegue de la función calidad (DFC) es una técnica de gestión de calidad[10] que traduce las necesidades del cliente en requisitos técnicos del software. DFC identifica 3 tipos de requisitos:

Requisitos Normales: Se declaran objetivos y metas para un producto o sistema durante las reuniones con el cliente, si estos requisitos están presentes el cliente quedará satisfecho. Ejemplos de requisitos normales podrían ser peticiones de tipo de presentación gráfica, funciones específicas del sistema y niveles definidos de rendimiento.

Requisitos Esperados: Estos requisitos son implícitos al producto y al sistema. Pueden ser tan fundamentales que el cliente no los declara explícitamente. Su ausencia sería motivo de una

insatisfacción significativa. Ejemplo de requisitos esperados son: Facilidad de interacción hombre-máquina, buen funcionamiento y fiabilidad general, y facilidad de instalación del software.

Requisitos Innovadores: Estas características van más allá de las expectativas del cliente y suelen ser muy satisfactorias. Por ejemplo: se pide un software procesador de texto con las características estándar. El producto entregado contiene ciertas capacidades de diseño de página que resultan muy validas y que no eran esperadas.

En la reunión con el cliente el despliegue de función se emplea para determinar el valor de cada función requerida para el sistema. El despliegue de información identifica tantos los objetos de datos como los acontecimientos que el sistema debe producir y consumir. Estos están unidos en las funciones.

Finalmente, el despliegue de tareas examina el comportamiento del sistema o producto dentro del contexto de su entorno. El análisis de valor es llevado a cabo para determinar la prioridad relativa de requisitos durante cada uno de los tres despliegues mencionados anteriormente. Esta técnica posee insatisfacciones ya que no son capturados todos los requisitos necesarios y algunos poseen ambigüedades. El DFC utiliza observaciones y entrevistas con el cliente y examina datos históricos (por ejemplo informes de problemas) para la actividad de recogidas de requisitos [13].

1.4.4. Principios para realizar la Gestión de Requisitos del software.

La GR es el conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y sus cambios en cualquier momento. O sea, básicamente, consiste en gestionar los cambios a los requisitos acordados, las relaciones entre ellos, las dependencias entre la Especificación de Requisitos del Software (ERS)[11] y otros documentos elaborados por el proceso de desarrollo de software.

En el proceso de desarrollo del software existen principios fundamentales para realizar la GR:

- El acuerdo de los requisitos es el puente entre el desarrollo de requisitos y la GR.
- La GR incluye todas las actividades para mantener la integridad, exactitud y difusión de los acuerdos de los requisitos durante la vida del proyecto.

En la GR se asegura la consistencia entre los requisitos y el sistema construido (o en construcción). Es una actividad necesaria porque:

- Los requisitos son volátiles:
  - Mutantes o cambiantes: sufren ligeras variaciones.
  - Emergentes: surgen al ir analizando el sistema en profundidad.
  - Colaterales: surgen como efecto de la inclusión de otros requisitos.
  - Por compatibilidad: se añaden para adaptar el sistema a su entorno, debido a que el entorno físico y organizacional cambian, así como las políticas, las reglas y los procesos del negocio, provocando cambios en el sistema.
- La propia existencia del sistema va a generar nuevos requisitos por parte de los usuarios.

La Gestión de Requisitos implica:

- Definir procedimientos que establezcan los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos.
- Cambiar los atributos de los requisitos afectados.
- Mantener la trazabilidad hacia atrás, hacia delante y entre requisitos.
- Controlar las versiones del documento de requisitos.

Los requisitos cambian y esto persiste a lo largo de la vida del sistema. Los cambios ocurren por:

- Cambios tecnológicos.
- Cambios en las estrategias o prioridades del negocio.
- Modificaciones en leyes y/o regulaciones.
- Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas.
- Porque cambió el problema que se estaba resolviendo.
- Porque los usuarios cambiaron su forma de pensar o sus percepciones.
- Porque cambió el ambiente de negocios.
- Porque cambió el mercado en el cual se desenvuelve el negocio.

Los cambios deben controlarse y documentarse. Por lo que, es esencial planear posibles cambios a los requerimientos cuando el sistema sea desarrollado y utilizado. Por tanto la GR del software, de su evolución es un proceso externo que ocurre a lo largo del ciclo de vida del proyecto.

Cambios a los requisitos involucra modificar el tiempo en el que se va a implementar una característica en particular, modificación que a la vez puede tener impacto en otros requerimientos. Por esto, la

gestión de cambios involucra actividades como establecer políticas, guardar históricos de cada requerimiento, identificar dependencias entre ellos y mantener un control de versiones.

1.4.5. Procedimiento de gestión de cambios en los requisitos.

Desde el inicio hay que establecer la conformación de la línea base de requisitos, como un canal simple para el control de cambios, que se podrá usar para "capturar" nuevos cambios.

Línea base de requisitos

Es el conjunto de requisitos funcionales y no funcionales que el equipo del proyecto se ha comprometido a implementar en una *release*[12] específica. Es una versión aprobada de la especificación de requisitos del software.

Los requisitos antes de entrar en la línea base deben ser sometidos a un procedimiento de revisión formal (con su aprobación para ser incluido en la línea base). Una vez entrado el requisito en la línea base cualquier cambio debe someterse al procedimiento de control de cambios.

Es adecuado que el documento de requisitos que se esté elaborando previo a entrar en la línea base esté sometido a un procedimiento de control de versión (para distinguir versiones borradores de versión aprobada).

Canal y control de cambios

En vista que las peticiones de cambios provienen de muchas fuentes, las mismas deben ser enrutadas en un solo proceso. Esto se hace con la finalidad de evitar problemas y conseguir estabilidad en los requerimientos.

La identificación del cambio se inicia desde el análisis de requisitos, nuevas necesidades del cliente, problemas operacionales; después se realiza el análisis del cambio y evaluación de costos, debiendo quedar claro cuántos requisitos se verán afectados y cuánto costará (tiempo y recursos); sólo después se toma la decisión de la implementación del cambio, que será documentado al ser realizado.

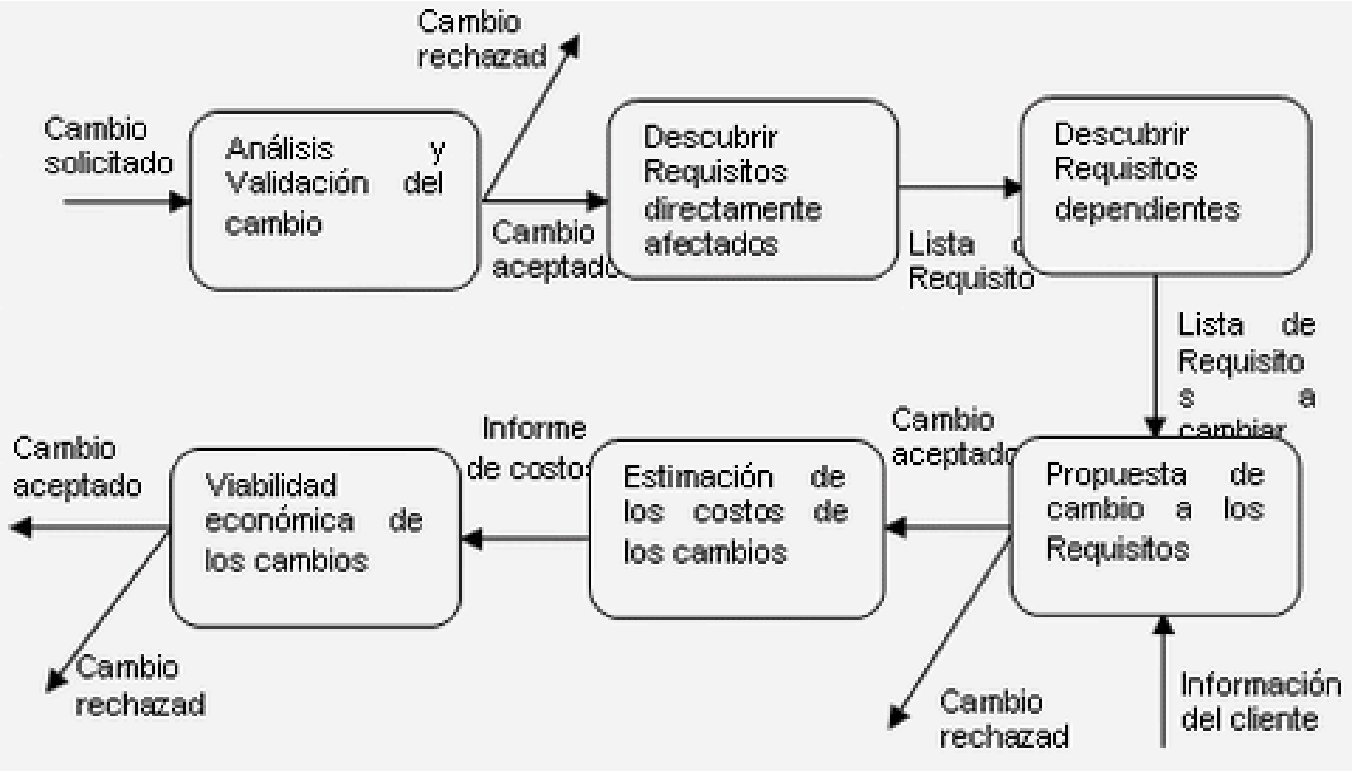


Figura 2: Proceso de control de cambios.

La gestión de los cambios se realiza de acuerdo a las prioridades, debiéndose identificar problemas y causas por los que se produce el cambio; analizar el impacto y el costo del cambio y finalmente, tomada la decisión, ejecutar (proceder con) el cambio.

Aspectos a considerar para la aprobación de un cambio solicitado:

- Impacto del cambio en el costo y funcionalidades del sistema.
- Impacto para el cliente y *stakeholders*[13] externos.
- Desestabilización potencial del sistema que pudiera ocasionar un cambio.

Las solicitudes de cambios, desde el momento en que son comunicadas hasta su implementación, transitan por diferentes estados en el que intervienen los implicados asumiendo diferentes roles. Obsérvese en la figura 3 y la tabla 3.

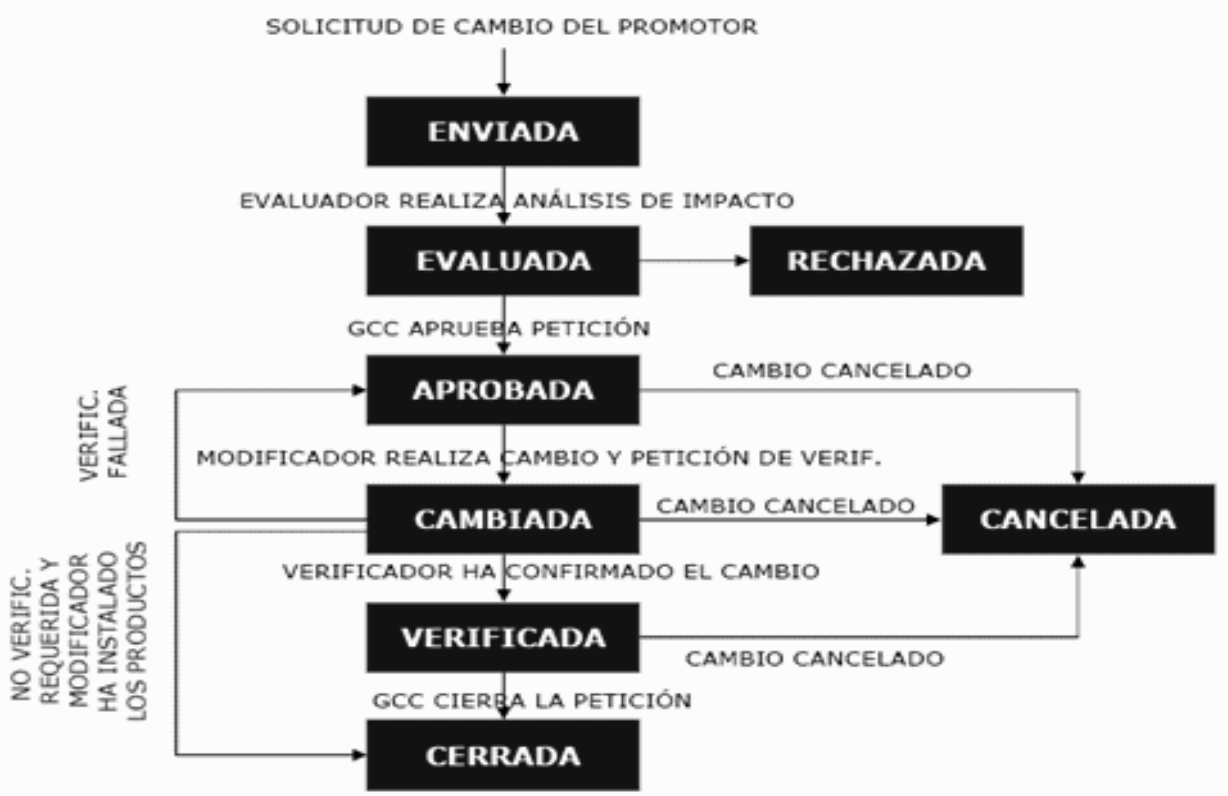


Figura 3: Posibles estados de una petición de cambio.

Rol	Descripción
Grupo de control de cambios (GCC)	Grupo de personas que deciden aprobar o rechazar las peticiones de cambios para un proyecto específico.
Promotor del cambio	Persona autorizada que solicita la petición de cambio de requisitos.
Evaluador	Persona que analiza el impacto de la petición de cambio (puedes ser técnico, martketing, cliente o combinación).
Modificador	Persona que realiza el cambio como consecuencia de una petición de cambio aprobada.
Verificador	Persona que determina si el cambio se ha realizado correctamente.
Validador	Persona del cliente que valida la implementación del cambio realizado.

Tabla 3: Posibles roles en el procedimiento de control de cambios.

En el Centro UCID, el grupo GCC que evalúa el impacto del cambio y aprueba el mismo es el Comité Central de Cambio compuesto por:

- Líder de proyecto.
- Arquitecto.
- Jefe de los analistas.
- Responsable de la Base de Datos.
- Gestor de Cambio.
- Gestor de Configuración.

El rol de Gestor de Cambio es encargado de evaluar, modificar, verificar y validar el cambio. Si el impacto del cambio es significativo es decir que afecte a otros proyectos se debe de llevar a nivel de jefatura para que sea evaluado y aprobado.

Ejecución de los cambios

Aprobado el cambio se procede a su implementación, de acuerdo a la fase del proyecto a que corresponda. En caso de que los cambios aprobados:

- Impliquen el desarrollo de un nuevo sistema, entonces será necesario comenzar un nuevo proceso de ingeniería de requisitos.
- Impliquen la implementación de nuevos requisitos, entonces será necesario comenzar por la actividad de elicitación o educción de requisitos.
- Afecten otras fases del proceso de desarrollo del proyecto, entonces se implementan en esas.

Trazabilidad

Los requisitos deben ser "rastreables": por su origen (¿quién lo propuso?); necesidad (¿por qué existe?); por su relación con otros requisitos; por su relación con elementos del diseño y/o la implementación. Esta información es útil para saber qué afecta un cambio en un requisito. Para resolver esta necesidad se usan las matrices para el seguimiento de los requisitos, entre ellas:

- Matriz de seguimiento de características (definidas por el cliente).
- Matriz de seguimiento de orígenes: identifica el origen de cada requisito.
- Matriz de seguimiento de dependencias: indica cómo se relacionan los requisitos entre sí.
- Matriz de seguimiento de subsistemas: vincula a los requisitos con los subsistemas (o módulos) que los manejan.
- Matriz de seguimiento de interfaces: muestra cómo los requisitos están vinculados a las interfaces internas y externas del sistema.

En la trazabilidad está definida como una técnica usada para "proveer una relación entre requisitos, el diseño y la implementación final del sistema" [14]. Se establece que "la trazabilidad da asistencia esencial en el entendimiento de las relaciones que existen entre y a través de los requisitos, el diseño y la implementación" [15].

Estas relaciones permiten mostrar que el diseño cumple con los requisitos funcionales y ayudan al reconocimiento temprano de aquellos requisitos que no son satisfechos por el diseño.

Control de versiones (evolución en el tiempo de la ERS).

Habitualmente la ERS necesitará ser modificada a medida que progresa el producto software, como resultado de los cambios aprobados en requisitos individuales o grupos de ellos. En la medida en que la especificación sea más completa, como consecuencia de que sus requisitos fueron especificados lo más completamente posible, menos versiones de la ERS habrá que producir.

Entre algunos de los beneficios que proporciona el control de versiones están:

- Prevenir cambios no autorizados.
- Guardar revisiones de los documentos de requerimientos.
- Recuperar versiones previas de los documentos.
- Administrar una estrategia de "releases".
- Prevenir la modificación simultánea a los requisitos.

Tener versiones de los requerimientos es tan importante como tener versiones del código, ya que evita tener requerimientos emparchados en un proyecto.

Para la trazabilidad y el control de versiones se realiza la identificación y almacenamiento de requisitos. Cada requisito y cada versión de ERS aprobada tendrán un identificador único.

Cuando la GR no se hace bien, de inmediato aparecen síntomas:

- Altos niveles de re-trabajo a lo largo del proyecto.
- Los requisitos se aceptan por el personal desde cualquier fuente que consideren fidedigna.
- Se incrementa de forma desmesurada las peticiones de requisitos.
- Incapacidad para probar que el producto cumple los requisitos aprobados.

¿Por qué se debe tener cuidado? Porque:

- La falta de acuerdo entre los afectados sobre cuáles son los requisitos reales, incrementa el tiempo y el costo del proyecto.
- Hay altas probabilidades de entregar un producto incompleto o incorrecto.
- Volver a revisar cambios en los requisitos, una y otra vez, es un derroche de recursos muy visibles para el cliente.

Buenas prácticas de la actividad de Gestión de Requisitos.

- Definir un proceso de control de cambios.
- Establecer un grupo (o comité) de control de cambios.
- Realizar análisis del impacto sobre los cambios.
- Crear líneas base y controlar las versiones de los requisitos.
- Mantener la historia de los cambios.
- Seguir el estado de los requisitos.
- Medir la volatilidad de los requisitos.
- Usar herramientas de Gestión de Requisitos.
- Crear matrices de trazabilidad de los requisitos.

El proyecto puede responder frente a petición de nuevos requisitos o petición de cambios en los requisitos de diferentes maneras:

- Diferir los requisitos de baja prioridad.
- Obtener recursos adicionales.
- Ordenar realizar más horas de trabajo (durante un período corto de tiempo y pagado).
- Retrasar el calendario para acomodarse a la nueva funcionalidad.
- Permitir reducir el nivel de calidad bajo la presión de mantener la fecha original.

Ninguna aproximación es universalmente correcta porque cada uno de los proyectos es diferente (características, presupuesto, calendario, recursos y calidad). Independientemente a cómo se responda a los cambios de los requisitos, lo que realmente importa es ajustar las expectativas y los compromisos cuando sea necesario.

La GR tiene como salidas los cambios aprobados y no aprobados, informes acerca del estado del proceso, de actividades o requisitos, resultados de análisis de matrices y otros.

1.5. Métricas de Calidad en el Software.

Las métricas del proceso del software pueden proporcionar beneficios significativos a medida que una organización trabaja por mejorar su nivel global de madurez del proceso. Sin embargo, al igual que todas las métricas, están pueden usarse mal, originando mas problemas de los que pueden solucionar.

Entonces, podemos definir las métricas de software o medidas de software como:
"La aplicación continua de técnicas basadas en las medidas de los procesos de desarrollo del software y sus productos, para producir una información de gestión significativa y a tiempo. Esta información se utilizará para mejorar esos procesos y los productos que se obtienen de ellos".

En la figura 4 se ilustra una extensión de esta definición para incluir los servicios relacionados al software como la respuesta a los resultados del cliente:

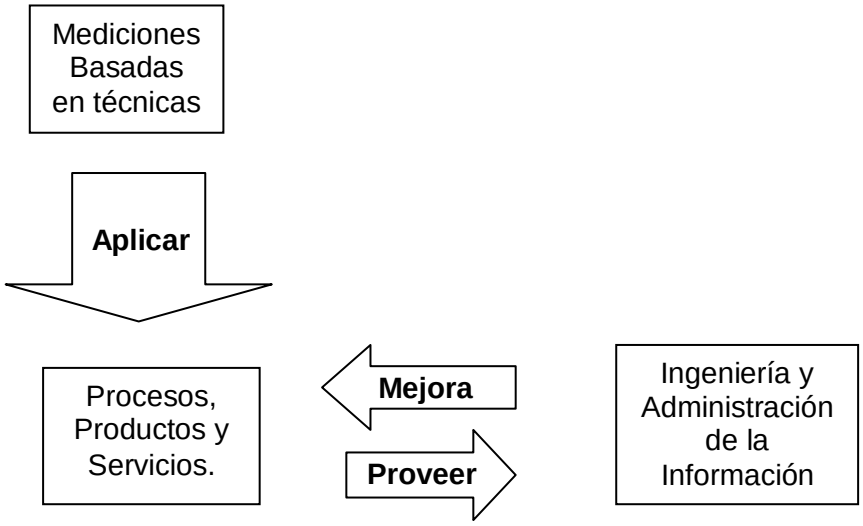


Figura 4: Servicios Relacionados al Software.

1.5.1 Las métricas en el proceso y dominio del proyecto.

Las métricas de proceso de software se emplean para fines estratégicos, y las métricas del proyecto de software son tácticas, éstas últimas van a permitir proporcionar al desarrollador de proyectos del software una evaluación al proyecto que sigue en continuo desarrollo, equivalentemente podrá ver los defectos que logren provocar riesgos a largo plazo (áreas problema); y observar si el área de trabajo (equipo) y las distintas tareas se ajustarán.

Las métricas de software nos aportan una manera de estimar la calidad de los atributos internos del producto, permitiendo así al ingeniero de software valorar la calidad antes de construir el producto, así

el tiempo invertido será identificando, examinando y administrando el riesgo, este esfuerzo merece la pena por muchas razones ya que habrá disminución de disturbios durante el proyecto, así mismo se podrá desarrollar una habilidad de seguir y controlar el proyecto y se alcanzará la seguridad que da planificar los problemas antes de que ocurran, además se conseguirá absorber una cantidad significativa del esfuerzo en la planificación del proyecto.

Las métricas del software:

- Ayudan a entender más acerca nuestros productos, procesos y servicios de software.
- Pueden ser empleadas para evaluar el software de nuestros productos, procesos y servicios con respecto a los estándares y metas establecidas.
- Proveen la información que nosotros necesitamos para controlar recursos y procesos utilizados en la producción de nuestro software.

1.5.2 Características de calidad. Sus métricas.

El modelo de la NC ISO/IEC 9126 presenta una variedad de requisitos de la evaluación, incluyendo, por ejemplo: el cliente podría evaluar un producto del software a partir de criterios predeterminados para las medidas externas de funcionalidad, confiabilidad, usabilidad y eficiencia; el responsable del soporte, servicio técnico o serviciador podría evaluar un producto del software usando las métricas de la mantenibilidad; la persona responsable de aplicar el software en los diferentes ambientes podría evaluar el producto de software usando métricas de portabilidad; un diseñador podría evaluar un producto del software a partir de valores predeterminados y usando medidas internas de cualquiera de las características de calidad.

- **Funcionalidad**: El grado en que el software satisface las necesidades.
- **Confiabilidad:** Cantidad de tiempo que el software está disponible para su uso.
- **Usabilidad:** Grado en que el software es fácil de usar.
- **Eficiencia**: Grado en que el software hace óptimo el uso de los recursos del sistema.
- **Facilidad de mantenimiento:** La facilidad con que una modificación puede ser realizada.
- **Portabilidad:** La facilidad con que el software puede ser llevado de un entorno a otro.

Funcionalidad

Las métricas para la medición de la característica funcionalidad.

Estas deben ser capaces de medir un atributo como es el comportamiento funcional del sistema en el cual el software está presente. Estas son:

Métricas de Idoneidad: Deben ser capaces de medir un atributo como es la ocurrencia de un funcionamiento insatisfactorio o la ocurrencia de una operación insatisfactoria. Un funcionamiento u operación insatisfactoria puede ser:

- Funcionamiento u operación que no se desempeña de la forma especificada en el Manual de usuario o la Especificación de Requisitos.
- Funcionamiento u operación que no provee una salida aceptable o razonable al tomar en consideración un objetivo especifico de las tareas del usuario.

Métricas de Precisión: Deben ser capaces de medir un atributo como es la frecuencia con que el usuario se encuentre con la ocurrencia de una falta de exactitud o de precisión, como puede ser:

- Resultados incorrectos o imprecisos causados por datos inadecuados; por ejemplo, un dato con pocos dígitos significativos para un cálculo de precisión; inconsistencia entre el procedimiento de operación actual y el descrito en el manual de operación.
- Diferencias entre el resultado actual y el razonablemente esperado producto de una tarea ejecutada durante la operación.

Métricas de Interoperabilidad: Deben ser capaces de medir un atributo como es el número de funciones o la ocurrencia de la menor incomunicación que involucre a datos y comandos o instrucciones que sean transferidos entre el producto de software y otros sistemas, otros productos de software u otros equipos a los cuales está conectado.

Métricas de Conformidad de la Funcionalidad: Deben ser capaces de medir un atributo como lo es el número de funciones con dificultades en la conformidad(o la ocurrencia de problemas de conformidad) con las regulaciones, normas u otras convenciones relacionadas, lo cual haga que el producto de software falle en adherirse a las mismas.

Confiabilidad

Las métricas para la medición de la característica confiabilidad.

Estas deben ser capaces de medir atributos relacionados con el comportamiento del sistema del cual el software forma parte durante la ejecución de las pruebas para indicar la magnitud de la confiabilidad, o sea, seguridad de funcionamiento del software durante la operación del sistema, con las que en la mayor parte de los casos no se distingue entre el software y el sistema. Ellas son:

Métricas de Madurez: Deben ser capaces de medir un atributo como la extensión de fallos en el software, causados por la ocurrencia de fallos existentes en el propio software.

Métricas de Tolerancia entre Fallos: Deben estar relacionadas con la capacidad del software de mantener un nivel de ejecución específico en caso de fallos de operación, o se infrinjan las interfaces específicas.

Métricas de Recuperabilidad: Deben ser capaces de medir aquellos atributos como el software y sistemas capaces de restablecer su nivel adecuado de ejecución y recuperar los datos directamente afectados en casos de fallos totales.

Usabilidad

Las métricas para la medición de la característica usabilidad

Estas miden la dimensión con que el software puede ser comprendido, estudiado, operado, atractivo y concordante con las regulaciones y guías relativas a la usabilidad. Ellas son:

Métricas de Comprensibilidad: Deben ser capaces de valorar como un nuevo usuario podría comprender:

- Si el software es idóneo para la aplicación a la cual lo destina, y como el software puede ser usado para una tarea en particular.

Métricas de Atracción: Deben ser capaces de evaluar la apariencia del software, y van a estar influenciadas por factores tales como el color en la pantalla y su diseño.

Métricas de Conformidad de la Usabilidad: Deben ser capaces de evaluar la adherencia del software a las regulaciones, normas, convenciones, guías y estilos relativos a la usabilidad.

Métricas de Cognoscibilidad y Operabilidad: Las métricas de cognoscibilidad (para medir el grado en que se puede ser estudiado) y las de operabilidad (para medir el grado en que puede ser implementado y operado) emplean métodos de aplicación eminente de usuarios y no son idóneas para el empleo por terceros en una evaluación de certificación, por lo que no se abordan en el presente trabajo.

Eficiencia

Las métricas para la medición de la característica eficiencia

Estas deben ser capaces de medir un atributo como es rendimiento o consumo de tiempo y el comportamiento en la utilización de los recursos del sistema, incluyendo al software, durante las pruebas y la implantación. Estas son:

Métricas de Rendimiento: Deben ser capaces de medir un atributo como es el crono comportamiento del sistema, incluyendo al software, durante las pruebas y la implantación.

Métricas de Utilización de Recursos: Deben ser capaces de medir un atributo como es el comportamiento en la utilización de los recursos del sistema, incluyendo al software, durante las pruebas y la implantación.

Métricas de Conformidad de la Eficiencia: Deben ser capaces de medir un atributo como es la adherencia del software a las regulaciones, normas, convenciones y guías relativas a la eficiencia.

Mantenibilidad

Las métricas para la medición de la característica mantenibilidad

Estas deben ser capaces de medir un atributo como es el comportamiento del serviciador, el usuario o el propio sistema, incluyendo el software, cuando el software es objeto de mantenimiento o modificación, durante las pruebas y el mantenimiento. Ellas son:

Métricas de Diagnóstico: Deben ser capaces de medir un atributo como es el esfuerzo del serviciador o el usuario y el gasto de recursos cuando se intente diagnosticar defectos o causas de fallos o para la identificación de partes a modificar.

Métricas de Flexibilidad: Deben ser capaces de medir un atributo como es el esfuerzo del serviciador o el usuario por medio de la medición del comportamiento del serviciador, el usuario o el propio sistema, incluyendo el software, cuando el software es objeto de introducción de determinada modificación.

Métricas de Estabilidad: Es la detección del comportamiento inesperado del sistema. Cuando el software es objeto de pruebas.

Métricas de Contrastabilidad: Es la medición del comportamiento del serviciador, el usuario o el propio sistema, incluyendo el software, cuando se intenta probar el software.

Portabilidad

Las métricas para la medición de las características portabilidad

Estas deben ser capaces de medir un atributo como es el comportamiento del operador y el sistema durante las actividades de la implementación y distribución del producto de software. Estas son:

Métricas de Instalabilidad: Deben ser capaces de medir un atributo como es el comportamiento del sistema y del serviciador o el usuario cuando el mismo está intentando instalar el software en un ambiente específico del usuario.

Métricas de Conformidad de la Portabilidad: Deben ser capaces de medir un atributo como es la adherencia del software a las regulaciones, normas, convenciones y guías relativas de la portabilidad.

1.5.3 Métricas de la calidad en el proceso de especificación de requisitos.

Para garantizar que nuestros proyectos vayan por el buen camino, se debe comenzar por la medición de la calidad de las ERS.

Existen ciertas cualidades deseables en cualquier especificación:

- **Completa:** describe toda la situación del cliente.
- **Consistente:** sin conflictos internos entre requisitos.
- **Correcta:** describe de forma precisa y exacta la situación y necesidades del cliente.
- **Modificable:** documentados de forma estructurada y accesible.
- **Ordenable:** los requisitos pueden ordenarse de acuerdo a su importancia o estabilidad.
- **Verificable:** debe ser sencillo determinar la cualidad éxito/fallo de cada requisito.
- **Trazable:** con el resto de elementos del ciclo de vida.
- **No ambigua:** cada requisito sólo se puede interpretar de una única forma.

Además las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o al menos interpretables, anotadas por importancia y estabilidad relativas, con su versión correspondiente, organizadas, con referencias cruzadas y especificadas al nivel correcto de detalle. Aunque muchas de las cualidades anteriores parecen ser de naturaleza cualitativa Davis [16], sugiere que todas pueden representarse usando una o más métricas.

1.5.4 Métricas de calidad aplicables a la Gestión de Requisitos.

Se puede generar una larga lista de características de la calidad del software: corrección, eficacia, portabilidad, mantenimiento, fiabilidad. A veces las características se solapan y entran en conflicto una con otras. Por ejemplo, incrementar la portabilidad, que es muy deseable puede dar lugar a una eficacia menor.

Dentro de este tipo de métricas, se ha tenido considerable atención en tres áreas:

- ♣ Corrección de los programas, medida como el número de defectos.
- ♣ Fiabilidad de software, calculada a partir del dato anterior.
- ♣ Mantenimiento de software, que se mide a partir de otro conjunto de métricas.

Métricas de Corrección

Un programa debe operar correctamente o proporcionará poco valor a sus usuarios. La corrección es el grado en el que el software realiza la función requerida. La medida más común de la corrección es el número de defectos *por KLDC*, en donde un defecto se define como una falta verificada de conformidad con los requisitos.

- ✓ **Métricas de Defectos.**

Se puede definir un defecto como la evidencia de la existencia de un error[14] en el software que produce un resultado incorrecto para un input válido. La medición de defectos no sólo se limita a contabilizar errores, sino que también incluye la contabilización del número de cambios tanto en diseño como en el código asociado a dicho error. Muchos de los errores que se descubren durante una fase han sido introducidos en fases previas, por lo tanto es conveniente diferenciar los costos de remover un defecto en la fase en que es introducido, con los costos de removerlo en una fase muy posterior. Mientras más tarde es removido, más caro. El ideal seria que:

$$d1 >> d2 >> d3.$$

Donde:

- ♣ d1: es el número de errores descubiertos durante la revisión de diseño y código.
- ♣ d2: es el número de errores descubiertos durante la revisión de prueba.
- ♣ d3: es el número de errores descubiertos durante la revisión de mantención.

- ✓ **Medición de remoción de defectos de software.**

Un aspecto de calidad de software que puede medirse en forma tangible y convincente es la medición de remoción de defectos. Esta métrica es uno de los parámetros fundamentales que debe ser incluida en cualquier programa de medición de software corporativo.

La frecuencia natural de medición de defectos es mensual, lo que se materializa en un informe. Este informe debe contener al menos la siguiente información:

- ♣ Defectos encontrados en revisiones e inspecciones.
- ♣ Defectos encontrados en prueba.
- ♣ Defectos encontrados e informados por los usuarios.

Otra información que se podría incluir:

- ♣ Defectos encontrados por producto y línea de producto.
- ♣ Defectos encontrados por región geográfica.
- ♣ Defectos encontrados por cliente.
- ♣ Defectos encontrados por segmento industrial.

Como ya se ha mencionado, los defectos deben comenzar a contarse lo más temprano posible, idealmente durante la fase de requerimientos.

- ✓ **Medición de la Eficiencia de la Remoción de Defectos.**

Una métrica de la calidad que proporciona beneficios tanto a nivel del proyecto como del proceso, es la eficacia de la eliminación de defectos (EED). En esencia EED es una medida de la habilidad de filtrar las actividades de la garantía de la calidad y de control al aplicarse a todas las actividades del marco de trabajo del proceso.

Cuando un proyecto se toma en consideración globalmente, EED se define de la forma siguiente:

$$EED = E / (E + D)$$

Donde E es el número de errores encontrados antes de la entrega del software al usuario final y D es el número de defectos encontrados después de la entrega. El valor ideal de EED es 1 es decir, no se han encontrado defectos en el software. De forma realista, D > 0, pero el valor de EED todavía se puede aproximar a 1. Cuando E aumenta (para un valor de D dado), el valor total de EED empieza

aproximarse a 1. De hecho, a medida que E aumenta, es probable que el valor final de D disminuya (los errores se filtran antes de que se conviertan en defectos). Si se utilizan como una métrica que proporciona un indicador de la habilidad de filtrar las actividades de la garantía de la calidad y de control, EED establece técnicas para encontrar todos los errores posibles antes de su entrega.

EED también se puede utilizar dentro del proyecto para evaluar la habilidad de un equipo en encontrar errores, antes de que pasen a la siguiente actividad estructural o tarea de ingeniería del software. Por ejemplo, la tarea del análisis de los requisitos produce un modelo de análisis que se puede revisar para encontrar y corregir errores. Esos errores que no se encuentran durante la revisión del modelo de análisis se pasan a la tarea de diseño (donde se pueden encontrar o no). Cuando se utilizan en este contexto, EED se vuelve a definir como:

$$EED_i = E_i / (E_i + E_{i+1})$$

En donde E_i es el número de errores encontrado durante la actividad de ingeniería del software i y E_{i+1} es el número de errores encontrado durante la actividad de ingeniería del software $i+1$ que se puede seguir para llegar a errores que no se detectaron en la actividad de la ingeniería de software i.
Es importante medir los defectos encontrados en cada fase de la serie de pasos en la remoción de defectos. Estos son:

- Revisión de requerimientos.
- Revisión de diseño.
- Inspección de código.
- Prueba de integración.
- Prueba de aceptación.

Para determinar la eficiencia en la remoción se necesita contar con los datos de detección de defectos por los usuarios de la aplicación, lo que implica recolectar datos durante todo el período de explotación del producto. Normalmente este análisis se realiza anualmente y determina la eficiencia en la remoción mediante la utilización de los datos de defectos detectados durante el primer año de la operación.

Métricas de Facilidad de Mantenimiento

El mantenimiento del software cuenta con más esfuerzos que cualquier otra actividad de ingeniería del software. La facilidad de mantenimiento es la habilidad con la cual se puede corregir un programa si se

encuentra un error, se puede adaptar si su entorno cambia, o mejorarlo si el cliente desea un cambio en los requisitos. No hay forma de medir directamente la facilidad de mantenimiento; por lo tanto, debemos utilizar medidas indirectas. Una métrica sencilla orientada al tiempo, es el tiempo medio entre cambios (TMEC) que lleva analizar la petición de cambio, en diseñar una modificación adecuada, en implementar el cambio, en probarlo y en distribuir el cambio a todos los usuarios.

Métricas de Integridad

Este atributo mide la habilidad del sistema para resistir ataques (tanto accidentales como intencionales) contra su seguridad. El ataque se puede realizar en cualquiera de los 3 componentes del software: programas, datos y documentos.

Para medir la integridad, se tiene que definir dos atributos adicionales: amenaza y seguridad. Amenaza es la probabilidad de que un ataque de un tipo determinado ocurra en un tiempo determinado. La Seguridad es la probabilidad de que se pueda repeler el ataque de un determinado tipo.

La integridad del sistema se puede definir como: Integridad = Σ [1−amenaza*(1−seguridad)]. Donde se suma la amenaza y la seguridad para cada tipo de ataque.

Métricas de Facilidad de Uso

La facilidad de uso es un intento de cuantificar la “amistad con el usuario” y se puede medir en función de cuatro características:

- ♣ Habilidad intelectual y/o física requerida para aprender el sistema.
- ♣ Tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema.
- ♣ Aumento neto en productividad (sobre el enfoque que el sistema reemplaza), medida cuando el sistema lo utiliza alguien moderadamente y eficiente.
- ♣ Valoración subjetiva (a veces obtenida mediante un cuestionario) de la disposición de los usuarios hacia el sistema.

Métricas de defectos informados por los clientes

¿Si se entrega un producto de software a un cierto número de clientes/usuarios y tiene un cierto número de defectos latentes, cuántos defectos serán detectados e informados por los clientes/usuarios en el primer año? ¿Cuántos en el segundo?

Se han observado dos reglas generales para la respuesta:

- El número de defectos encontrados se correlaciona directamente con el número de usuarios; mientras más usuarios, más defectos se encuentran.
- El número de defectos encontrados se correlaciona inversamente con el número de defectos presentes; mientras más defectos, menos defectos se encuentran.

La primera regla es auto-exploratorio, e implica que a mayor número de usuarios, mayor es el porcentaje de defectos detectados durante el primer año. La segunda es más difícil de entender, pero se basa en el fenómeno de que software muy defectuoso deja de usarse rápidamente, o sea usa muy poco, por lo que se estira el período en que los errores se detectan y corrigen.

1.6. Situación existente en los proyectos del centro UCID.

En el centro UCID existen varias categorías de proyectos como: portal, gestión, software base, multimedia, entre otros. Estos están basados en la metodología RUP. De los 28 proyectos existentes ha sido asignado Contabilidad Material para la aplicación de métricas de calidad para la GR, el cual se encuentra en estado avanzado, en la etapa de implementación de varios módulos y con otros en continuación, surgido a medida que el cliente lo solicitó. Su dificultad es que no se rige adecuadamente por la metodología existente, por lo que no emplea todas las actividades necesarias y suficientes para el procedimiento de la GR. En la fase de inicio cuando se empezó a realizar el flujo de trabajo de Requerimiento, se capturaron los requisitos y fueron documentados en las plantillas del Expediente de Proyecto, destinadas por el laboratorio de calidad del centro, pero a lo largo del ciclo de vida del software no se controlaron ni se documentaron los cambios realizados en los requisitos por lo que no se podía llevar a cabo la aplicación de métricas de calidad. Por lo antes expuesto se ha decidido abarcar todos los proyectos pertenecientes al centro con el objetivo de proponer un procedimiento de GR aplicándole métricas de calidad.

1.7. Conclusiones.

Las métricas constituyen una base objetiva para poder planificar y controlar de forma realista la Gestión de Requisitos, minimizando el costo de los recursos dedicados a los proyectos en este flujo. Al mismo tiempo facilita la extracción de datos sobre la experiencia de los productos realizados convirtiéndolos en valiosa información para la toma de decisiones relativa a los proyectos nuevos o en curso.

En este capítulo se fundamentó sobre los temas de calidad, requisitos y métricas. Hasta el momento se han analizado un conjunto de factores cualitativos para la “medición” de la calidad del software demostrando que al aplicar métricas a la Gestión de Requisitos se obtienen importantes beneficios:

- Ayuda a orientar sobre las áreas del problema.
- Mejora la calidad del producto.
- Incrementa la productividad del equipo de desarrollo.
- Mejora la estimación y planificación del proyecto.
- Mejora la gestión del proyecto.
- Incrementa la satisfacción del cliente.
- Incrementa la visibilidad del proceso de software permitiendo profundizar en la planificación, control, gestión y mejora.

CAPÍTULO II: PROPUESTA DE MEJORA DEL PROCEDIMIENTO.

2.1. Introducción.

La relativa corta vida de la disciplina de la Gestión de Requisitos y el afán de la comunidad en resolver los problemas afines a ella, ha generado un sin número de métodos, metodologías, procesos y herramientas que en realidad no han llegado a resolver de manera satisfactoria todos los inconvenientes que se presentan en esta fase de cualquier proyecto de desarrollo. Con tal objetivo se ha valorado el uso de métricas de calidad como técnica que puede ayudar en la mejora del proceso de GR pues descubre si los cambios en el proceso mejoran la eficiencia de este.

En este capítulo se presenta como propuesta, un procedimiento para la GR utilizando como base lo establecido por RUP y teniendo como objetivo el aseguramiento de la producción de software de calidad dentro de plazos y presupuestos predecibles. Se prevé una mejor práctica para la verificación de la calidad del software y el control de cambios por lo que es necesario gestionar los requisitos de los elementos del proyecto y sus componentes e identificar posibles inconsistencias.

2.2. Análisis de los requisitos.

Tanto el desarrollador como el cliente tienen un papel activo en la IR del software, un conjunto de actividades que son denominadas análisis como tarea que cubre el hueco entre la definición del software a nivel de sistema y el diseño del mismo (Figura 5). El análisis permite al ingeniero del sistema especificar las características operacionales del software (función, datos, rendimientos), indica la interfaz con otros elementos del sistema y establece las restricciones que debe cumplir.

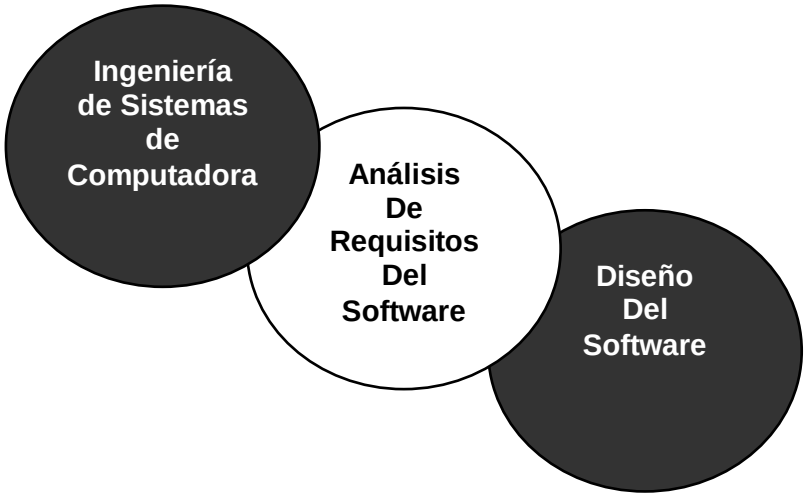


Figura 5: Análisis como puente entre la ingeniería y el diseño del software.

El análisis de requisitos del software puede dividirse en 5 áreas de esfuerzo:

1. Reconocimiento del problema.
2. Evaluación y Síntesis.
3. Modelado.
4. Especificación.
5. Revisión.

Inicialmente el analista, estudia la especificación del sistema (si existe alguna) y el plan del proyecto del software. Es importante entender el software en el contexto de un sistema y revisar el ámbito que se empleó para generar las estimaciones de la planificación. A continuación se debe establecer la comunicación para el análisis de manera que se garantice un correcto reconocimiento del problema. El objetivo del analista es identificar los elementos básicos del problema tal y como los percibe el cliente.

Para la ingeniería de requisitos, Davis [17], sugiere un conjunto de principios directrices:

- ♣ Entender el problema antes de empezar el modelo de análisis.
- ♣ Desarrollar prototipos que permitan al usuario entender como será la interacción hombre-máquina.
- ♣ Registrar el origen y la razón de cada requisito.
- ♣ Usar múltiples planteamientos de requisitos.
- ♣ Dar prioridad a los requisitos.
- ♣ Trabajar para eliminar la ambigüedad.

2.3. Estudio del procedimiento aplicado por los proyectos de la UCID para desarrollar la Gestión de Requisitos.

A pesar de la importancia que tiene la puesta en práctica de un procedimiento que viabilice la gestión de los requisitos en proyectos pertenecientes al MINFAR, no es una práctica común el que sean desarrolladas las principales actividades correspondientes a este flujo de manera que constituyan una base para la toma de decisiones futuras. La gran mayoría de los proyectos no documentan los cambios realizados en la captura de requisitos; aspectos que pudieron ser confirmados en encuesta realizada para recoger las experiencias de los proyectos desarrollados en esta área de la universidad. (Ver Anexo 1).

En el centro existen 28 proyectos en total para un 100%, todos nacionales, de los cuales fueron encuestados 15 para un 53.6% del total (Figura 6).

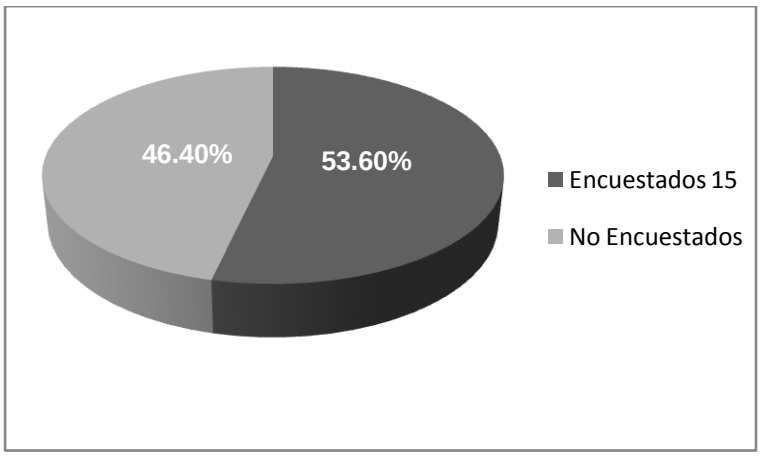


Figura 6: Gráfica de por cientos de proyectos encuestados.

De las metodologías de desarrollo:

Metodología	No. de proyectos	Metodología Ágil	Metodología Robusta	Proceso Unificado del Desarrollo (RUP)
Si utilizó	13---86.6%	_	13---86.6%	13---86.6%
No utilizó	2----13.4%	_	_	_

Tabla 4: Uso de las metodologías de desarrollo.

Existe un proyecto que no desarrolla ningún tipo de metodología, pues su líder expresa que es un proyecto atípico y realizado con rapidez, sin tiempo alguno para realizar la ingeniería de software. A pesar de estos inconvenientes, sus integrantes tienen conocimiento de la misma y se muestran de acuerdo con los beneficios que reporta su aplicación.

Del procedimiento de Gestión de Requisitos:

El procedimiento de GR aplicado por los proyectos encuestados, según las estadísticas, se comporta de la siguiente manera:

Lo utilizaron	Cantidad de proyectos	Por ciento
Sí	10	66.6 %
No	5	33.4%

Tabla 5: Uso del procedimiento para la Gestión de Requisitos.

Este procedimiento, descrito por RUP, cuenta con un conjunto de actividades fundamentales. En el siguiente gráfico (Figura 7), se puede observar cuáles son las actividades que más se desarrollan como Entender las necesidades de los stakeholders. Dentro de las que menos se realiza está Administrar los cambios de requerimientos, actividad fundamental que se debe llevar a cabo.

Numeración	Actividades	No. de proyectos	
		Si realizan	No realizan.
1ra	Análisis del problema	11	4
2da	Entender las necesidades de los stakeholders	12	3
3ra	Definir el sistema	8	7
4ta	Administrar el alcance del sistema	9	6
5ta	Refinar la definición de sistema	10	5
6ta	Administrar los cambios de requerimientos	4	11

Tabla 6: Cantidad de proyectos que desarrollan las actividades de GR.

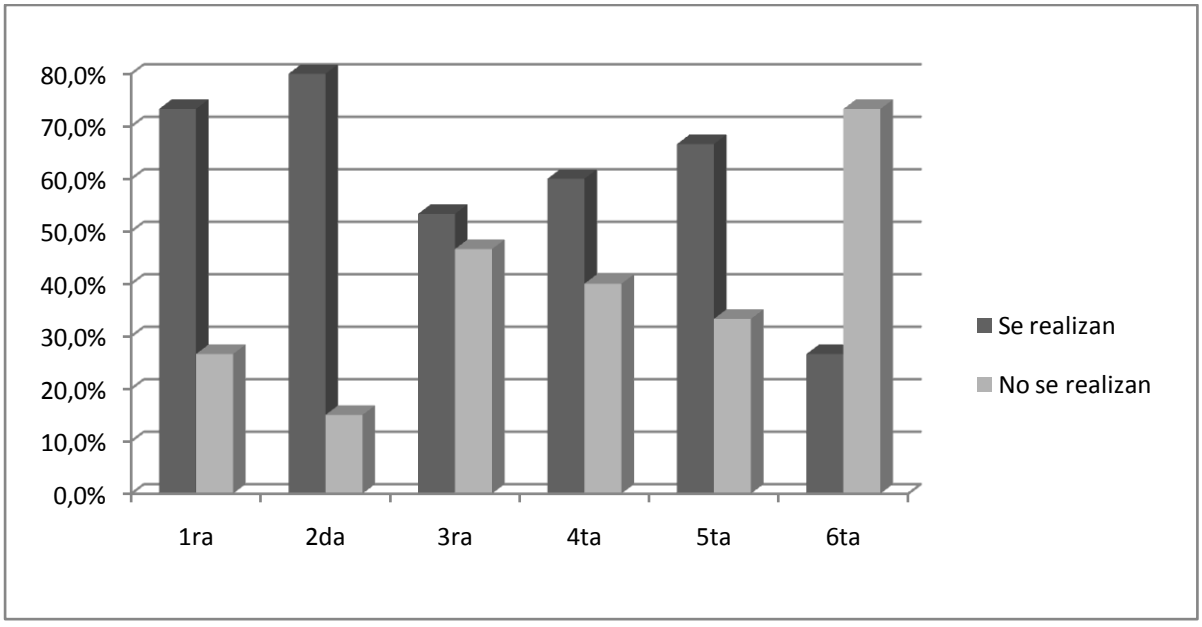


Figura 7: Por cientos de actividades realizadas durante la GR.

Después de la captura de requisitos inicial, en el proyecto se realizan cambios que quedan registrados de la siguiente manera:

- Realizaron cambios un total de 11 proyectos para un 73.3%.

- ♣ No realizaron cambios un total de 4 proyectos para un 26.7%.

Para tener una idea de lo anteriormente explicado, se presenta un esquema basado en la correspondencia de los flujos de trabajo con estos cambios. El análisis y diseño se muestra como la etapa de trabajo donde más cambios se realizan.

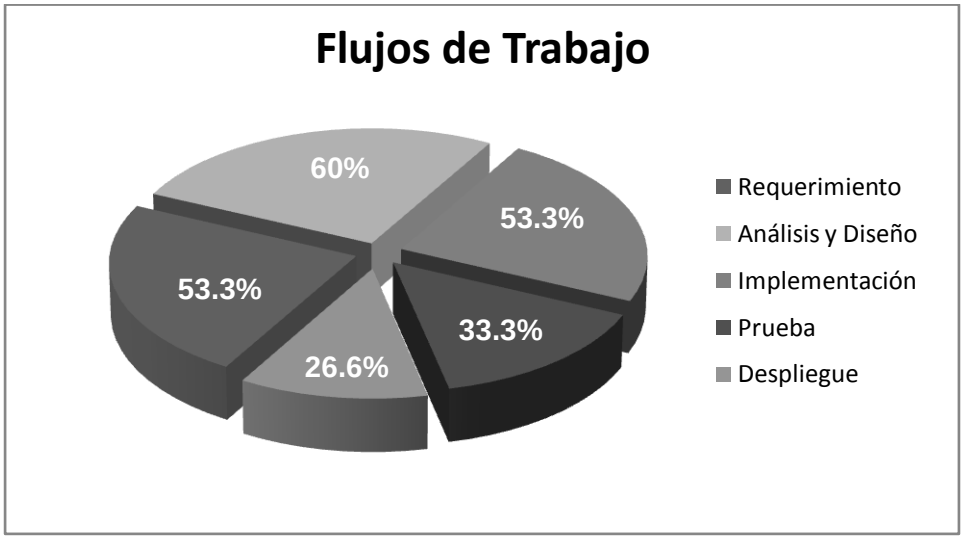


Figura 8: Cambios en cada flujo de trabajo.

Las causas que conllevaron a que se realizaran estos cambios en los flujos de trabajos fueron diversas, la encuesta arrojó los resultados siguientes:

Causas	No. de proyectos	Por ciento
Petición de cambios en los requisitos	9	60%
No se entrevistó a la persona adecuada	1	6.6%
Insatisfacción con el software	3	20%
Incorrecta comunicación con el cliente	4	26.6%
Otras	4	26.6%

Tabla 7: Causas de los cambios en los requisitos.

Otras causas:

- ♣ Cambios no controlados y sin consultar con los desarrolladores.
- ♣ Cambio en el liderazgo del proyecto.
- ♣ Falta de experiencia del tema en el centro.

De acuerdo a la encuesta aplicada a los proyectos de la Unidad de Compatibilización, Integración y Desarrollo de software para la defensa, se puede decir que de los 15 proyectos encuestados solo 4 documentan los cambios en los requisitos (para un 26.7%). Se identifica entre las causas fundamentales de este hecho que el tiempo de realización de un producto es muy limitado por lo que no alcanza para escribir y registrar los requisitos capturados, pues se efectúan cambios constantemente, es un trabajo contra reloj. Igualmente se cometen errores pues se sobrescriben las plantillas de Especificación de Requisitos y por lo tanto no se dejan plasmado los cambios que se realizaron con anterioridad, existe poca experiencia sobre el tema pues no se tiene a la persona encargada de documentarlos, de la misma forma no es muy rigurosa la exigencia a los líderes de proyectos alrededor del cumplimiento de esta actividad. Tampoco existe el documento de control de cambios; ni la definición exacta para realizar el proceso.

Al finalizar la aplicación de la encuesta se pudo medir el grado de dificultad en que se encuentran los proyectos del MINFAR como punto de partida hacia el mejoramiento del procedimiento de Gestión de Requisitos con la metodología existente (RUP).

2.4. Modelo del proceso de Gestión de Requisitos.

Existen diferentes modelos que representan la Gestión de Requisitos, creados por diferentes autores como Suzanne y James Robertson, Pohl, Sommerville y Sawyer, Amador Durán, entre otros. Todos estos modelos encierran un conjunto de procesos y actividades similares para garantizar que se desarrolle el software con una alta calidad.

En este apartado se va a presentar el modelo que fue creado por el proyecto SWEBOK (Software Engineering Body of Knowledge) de la IEEE para producir un cuerpo de conocimiento sobre ingeniería de software que siente las bases de dicha ingeniería como una profesión. Dicho modelo lleva el mismo nombre del proyecto y se escogió por la claridad de los procesos que describe (Figura 9). [18]

Las siguientes etapas son componentes fundamentales dentro de la actividad de captura de requisitos para cualquier proyecto: Elicitación, Análisis y Negociación, Documentación de los requisitos, y Validación. Analizando las necesidades de los usuarios, la información del dominio, sistema de información existentes, estándares y regulaciones.

La Figura 9 representa las actividades básicas que propone el modelo presentado, sus relaciones y los artefactos que se obtienen al finalizar el proceso.

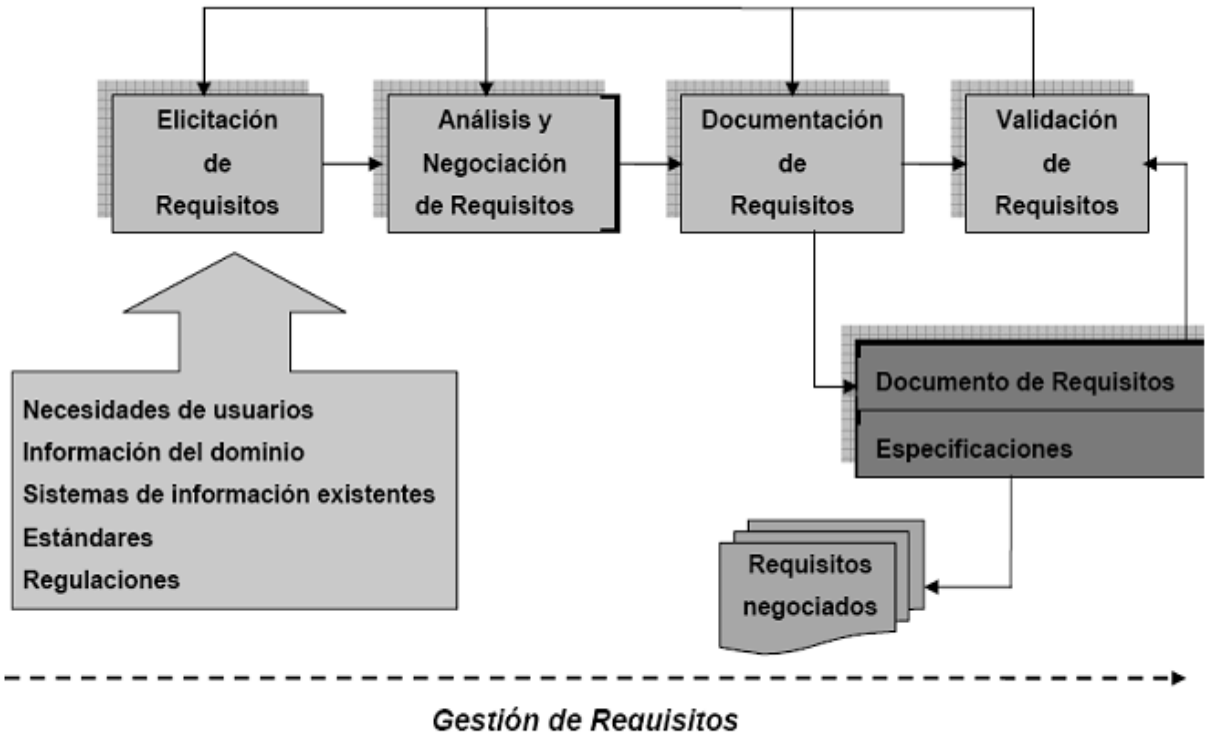


Figura 9: Modelo SWEBOK del proceso de Gestión de Requisitos.

Elicitación de los requisitos: Actividad de la IR en la cual se estudia el dominio del problema y se interactúa con los clientes y usuarios para obtener y registrar información sobre sus necesidades [19]. Antes de identificar los requisitos que el sistema debe cumplir, es conveniente conocer el ambiente y los procesos que se desarrollan dentro de la organización donde el sistema a construir va a ofrecer sus servicios, ésta es la etapa para recolectar y obtener toda la información posible y necesaria para modelar la organización en estudio. Se propone en esta etapa la obtención del Modelo de Negocio, el cual describe los procesos de negocios de la organización, especificando sus datos, tareas, roles, agentes y reglas, información a partir de la cual se identifican los primeros requisitos candidatos a ser cubiertos por el sistema a desarrollar.

Análisis y Negociación de los requisitos: Actividad de la IR en la cual se estudia la información extraída en la etapa previa para identificar la presencia de áreas no detectadas, requisitos contradictorios y peticiones que aparecen como vagas e irrelevantes [20]. En esta etapa se encuentra el Modelo de Dominio, que representa los objetos del negocio y del contexto y los eventos del entorno del sistema a construir, se clasifican y analizan los requisitos encontrados en la primera aproximación y se negocian

con el cliente para verificar los puntos de acuerdo y entendimiento de sus necesidades; es posible también, rediseñar los procesos del negocio para mejorarlos o adaptarlos para su sistematización o automatización cuando esté construido el sistema.

Documentación de los requisitos: Actividad de la IR en la cual se realiza una descripción formal de los requisitos que finalmente el sistema a construir va a cumplir. Se documentan los requisitos elicitados y negociados en tantas notaciones como sean necesarias para que todos los participantes los entiendan.

Validación de los requisitos: Actividad de la IR en la que clientes y usuarios, junto con la ayuda de los ingenieros de requisitos y otros evaluadores, revisan los productos obtenidos en etapas anteriores para comprobar que realmente reflejen sus necesidades, que definen el producto deseado y que están descritos de la manera correcta [21]. Una vez aprobados los requisitos por parte del cliente, se procede a actualizar el repositorio de requisitos del grupo desarrollador.

2.5. Actividades de la propuesta de procedimiento para la Gestión de Requisitos, basándose en la metodología RUP.

El procedimiento que se muestra a continuación tiene la facilidad de adaptarse a cualquier entorno de desarrollo, se puede adecuar a otras metodologías (ágiles y robustas) siempre y cuando no se altere el orden de las actividades y se lleve a cabo con todas las técnicas, plantillas, métricas de calidad y herramienta del procedimiento, teniendo en cuenta, en el flujo de trabajo Requerimiento, los roles y artefactos de la metodología a la que se quiere emigrar.

Para realizar el procedimiento lo primero es conocer los pasos del mismo representado en un diagrama de actividades en la figura 10:

1. Análisis del problema.
2. Entender las necesidades de los stakeholders.
3. Definir límites del sistema.
4. Administrar alcance del sistema.
5. Refinar definición del sistema.
6. Administrar cambios en los requisitos.

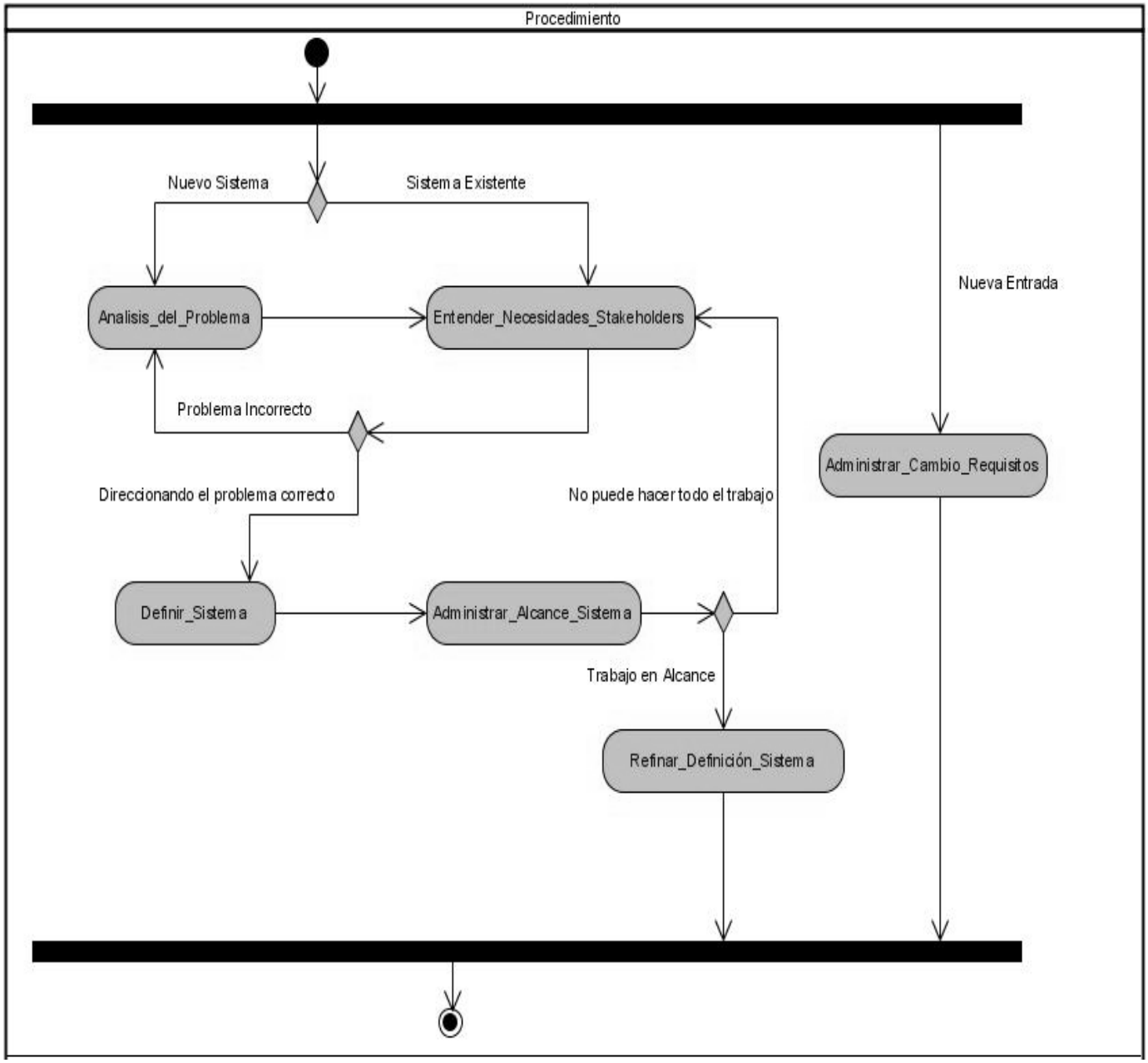


Figura 10: Diagrama de actividades del procedimiento.

En el centro UCID existen artefactos que son comunes para los flujos de trabajo. La propuesta acepta esta forma de construir los artefactos y plantea una mejora para el procedimiento que establece RUP para la GR, adaptándolo a las condiciones del centro. Para ello se ha omitido una actividad y una serie de artefactos pues se consideraron innecesarios para este flujo. Además existen otros no generados, mas se tienen en cuenta en otras plantillas, para realizar las actividades de la GR.

Artefactos comunes:

- Plantilla Documento Visión realizada en la Gestión de Proyecto, se encuentra en las plantillas de los restantes artefactos se tratan como alcance.
- Plantilla de Documento de la Arquitectura del Software realizada en Análisis y Diseño, es un artefacto que se completa a lo largo del desarrollo del software.
- Plantilla de Lista de Riesgos realizada en la Gestión de Proyecto, es un artefacto de entrada en este flujo de trabajo.
- Plantilla Glosario de Términos realizada en Soporte, se encuentra en las plantillas de los restantes artefactos como definiciones, acrónimos y abreviatura, esta última teniendo vital importancia ya que se resume todos aquellos conceptos de difícil comprensión.

Actividad omitida:

- Captura del vocabulario común: los artefactos que genera esta actividad se tienen en cuenta durante todo el procedimiento y no en una parte de este.

Artefactos omitidos:

- Especificaciones Suplementarias: es un documento de apoyo que solo contiene los requisitos no funcionales, existiendo la Especificación de Requisitos del Software siendo un artefacto más completo (Ver Anexo 7).
- Plan de iteración: se llevan a cabo varias iteraciones que se documentan en las plantillas de cada artefacto.

Artefactos no generados:

- Reglas del Negocio: se encuentra contenida dentro de la plantilla Modelo del negocio del flujo de trabajo de Negocio.
- Atributos de Requisitos: lo contiene como elemento la plantilla de Especificación de Requisitos del Software.
- Prototipo de Interfaz de Usuario: está contenido en la plantilla de Casos de Uso Detallados (Ver Anexo 5).

Las Reglas del Negocio, aunque es un artefacto no generado dentro del procedimiento, se tienen en cuenta durante el desarrollo de todas las actividades de la GR, ya que forman parte de los requisitos, le impone condiciones al futuro desarrollo del proceso de software. Además, con solo tener las reglas del negocio, en muchos casos se puede obtener una idea más completa, por parte del equipo de

desarrollo, del producto que se desea obtener.

En el procedimiento que se propone a continuación existen diferentes roles responsables de realizar las actividades en las que se generan artefactos de entrada o de salida.

Tipo de artefactos:

- Salida: Son los que se crean en la actividad.
- Entrada: Son los que se tiene en cuenta para elaborar los artefactos de salida.
- Ambas: Se actualiza el documento.

2.5.1. Análisis del problema.

En RUP esta actividad se inicia con la identificación de Actores y Casos de Uso del Sistema. El propósito del análisis del problema en este flujo de trabajo es estudiarlo para la futura informatización. Se definen como puntos principales:

- Entender el problema.
- Identificar los stakeholders.
- Definir límites del sistema.
- Identificar restricciones.

En el centro UCID el análisis del problema se inicia con el estudio del mismo por parte del analista del sistema, para cerciorarse que tanto clientes como desarrolladores están seguros de entender las necesidades planteadas. Para evitar equivocaciones es importante coincidir en cuanto a la terminología que se usará a lo largo del proyecto.

En el análisis del problema las actividades que se van a realizar y los artefactos que se generan son:

Rol	Actividad	Artefacto	Tipo de Artefacto
Analista del Sistema	Desarrollar el Plan de Gestión de Requisitos.	Plan de Gestión de Requisitos	Salida
	Desarrollar Documento Visión.	Requisitos de los Stakeholders	Salida
	Encontrar actores y casos de uso.	Modelo de Caso de Uso	Salida

Tabla 8: Relación rol, actividad y artefacto durante el análisis del problema.

En los proyectos de la UCID, el Plan de Gestión de Requisitos se usa para proporcionar una guía en la realización de esta actividad, incluye los artefactos que se deben generar, los tipos de requisitos que

deben manejarse para el proyecto, los atributos de estos requisitos y el acercamiento a su trazabilidad (Ver Anexo 2). Este plan hace referencia al Documento de Cambios en los Requisitos la cual se creó para documentar el seguimiento que se efectúe (Ver Anexo 6).

La plantilla Requisitos de los Stakeholders es uno de los artefactos que se propone se realice en el centro pues ofrece preguntas diseñadas para conseguir una comprensión de las necesidades de todos los involucrados. El uso de este artefacto proporciona al desarrollador o analista conocimiento del problema a resolver a través de la exploración de la funcionalidad, utilidad, fiabilidad, rendimiento y requisitos de soportabilidad para la aplicación. (Ver Anexo 3).

El Modelo Caso de Uso del Sistema es una plantilla que contiene información sobre las condiciones que debe cumplir el mismo: precondiciones, post-condiciones, requisitos especiales y escenario, son algunos de los elementos que se recogen en ella. En este modelo se hace una especificación de las propiedades textuales del caso de uso (Ver Anexo 4).

Esta actividad debe realizarse varias veces (según se considere necesario) durante las fases Inicio y Elaboración, con el objetivo de manejar cambios inevitables de los requisitos que ocurrirán en el proyecto para asegurar que se continúa dirigiendo el problema de la forma correcta.

Una vez analizado el problema y como método para obtener una buena captura de requisitos se propone realizar de manera conjunta las siguientes técnicas.

Técnicas para la captura de requisitos.

Para hacer un procedimiento que pueda ser aplicado durante la GR es preciso, como primer paso, saber que técnica emplear para obtener una buena captura de requisitos. Con el estudio realizado en el Capítulo I de la investigación donde se explicaron estas técnicas y se analizó el método que utilizan los proyectos del centro para la captura de requisitos, se puede decir que con la realización de una sola técnica no es suficiente para catalogar de efectivo el proceso de captura de requisitos. Teniendo en cuenta lo antes mencionado se ha recomendado realizar, siguiendo el orden propuesto, los siguientes pasos:

1. Entrevistas o reuniones.

Los desarrolladores tienen la oportunidad de reunirse con los clientes o usuarios para recopilar, mediante el sistema preguntas y respuestas, toda la información que necesiten para entender el problema en cuestión y procurar el logro del objetivo. Las preguntas ayudarán a romper el hielo e

iniciar la comunicación. Esta técnica es la más usada y efectiva para la captura de requisitos, en ella se tiene la posibilidad de dialogar con el cliente y realizarle preguntas concretas e importantes para la realización del software.

2. Técnica para facilitar las especificaciones de una aplicación.

Esta técnica, según su nombre lo indica, facilita las especificaciones de una aplicación. Una vez que el desarrollador sabe lo que desea el cliente, entonces elaboran los requisitos, de forma independiente. Después, en una breve reunión, se expone los requisitos capturados. Las pautas básicas para esta reunión son las siguientes:

- La reunión se celebra en un lugar neutral y acuden tantos los clientes como los desarrolladores.
- Se sugiere una agenda lo suficientemente formal como para cubrir todos los puntos importantes, pero lo suficientemente informal para animar el libre flujo de ideas.
- Se usa un mecanismo de definición (que pueden ser hojas de trabajo, gráficos o carteles).
- El objetivo es identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos de la solución en una atmósfera que permita alcanzar el objetivo.
- Se realiza el despliegue de la función calidad para alcanzar resultados satisfactorios con la mayor cantidad de requisitos posibles.

3. Despliegue de la función calidad.

Es una técnica de gestión de calidad que traduce las necesidades del cliente en requisitos técnicos del software. Se realiza en las reuniones o entrevistas con los clientes, se definen tres tipos de requisitos: normales, esperados e innovadores, estos últimos se emplea para determinar el valor de cada función requerida para el sistema.

Finalmente, el despliegue de tareas examina el comportamiento del sistema o producto dentro del contexto de su entorno y en conjunto con las dos técnicas anteriores se logran extraer los requisitos necesarios para el desarrollo del software.

2.5.2. Entender las necesidades de los stakeholders.

El principal objetivo de esta actividad son las entrevistas y talleres de requisitos donde se puede iniciar discutiendo los requisitos funcionales del sistema refiriéndose a sus casos del uso y actores. Los requisitos no funcionales se documentan en la Especificación de Requisitos del Software, teniendo en cuenta que en el Centro UCID son estándares para todos los proyectos.

Típicamente, esta actividad es realizada durante las iteraciones de las fases Inicio y Elaboración, sin embargo las demandas de los stakeholders adicionales continuarán siendo recogidas a lo largo del proyecto por la vía del Documento de Cambios en los Requisitos, artefacto propuesto para garantizar la trazabilidad de los requisitos mediante una matriz de seguimiento. Su realización es de gran importancia en el centro UCID debido que no se lleva un control de los requisitos. Por la importancia que tiene el entendimiento de las necesidades de los involucrados por parte del equipo de desarrollo, se propone esta como una actividad fundamental dentro del procedimiento. El objetivo principal es recoger la información relacionada con el producto deseado. Para su desarrollo se propone tratar como puntos principales:

- ♣ Identificar Casos de Uso.
- ♣ Realizar el diagrama de Casos de Uso.

La colección de los requisitos de los stakeholders puede considerarse como una "lista de deseos" que se usará como entrada para definir los rasgos del sistema descritos en la visión que maneja la Especificación de los Requisitos del Software, en el Modelo Caso de Uso del Sistema y la plantilla de Casos de Uso Detallados. En esta actividad el rol Analista del Sistema realiza las actividades que aparecen a continuación en las que se generan los artefactos señalados:

Rol	Actividad	Artefacto	Tipo de Artefacto
Analista del sistema	Desarrollar Documento Visión.	Plan de Gestión de Requisitos	Entrada
	Gestionar Dependencias	Modelo de Caso de Uso	Entrada y Salida
	Encontrar actores y casos de uso	Especificaciones de Requisitos del Software	Salida
	Elicitación de Requisitos de stakeholders	Requisitos de los Stakeholders	Entrada y Salida
		Documento de Cambios en los Requisitos	Salida

Tabla 9: Relación rol, actividad y artefacto en Entender las Necesidad de los Stakeholders.

Con el objetivo de cubrir la mayor cantidad de vías por las que se pueden capturar los requisitos, se propone una Lista de Auto-chequeo para el desarrollador (Ver Anexo 8).

Plantilla de Auto-chequeo para el desarrollador.

Generalmente cuando se van a capturar los requisitos se realizan una serie de preguntas al cliente para recoger los requisitos funcionales y no funcionales, a veces por mala comunicación con el cliente

y/o usuario no se recoge la información necesaria para el buen desarrollo del software.

La lista de auto-chequeo que se propone tiene como objetivo que el desarrollador encargado de realizar la captura de requisitos se formule una serie de preguntas para que ésta se desarrolle con la calidad requerida. Estas preguntas están elaboradas de forma clara, precisa y asequible con el propósito de obtener la mayor cantidad de requisitos.

2.5.3. Definir el sistema.

El propósito de definir el sistema es aproximarse en el alcance de los requisitos perfilándolos detalladamente para el sistema. Posee como tareas principales: describir y seguir la traza de los requisitos durante la comprensión de las necesidades de los stakeholders, gestionar el Alcance del Sistema y el Refinamiento de la Definición del Sistema. Se definen como puntos principales:

- ♣ Plantilla de Modelo de Casos de Uso.
- ♣ Propiedades del actor y Casos de Uso.
- ♣ Interacción entre el Actor y el Caso de Uso.

El Analista del sistema realiza las siguientes actividades y da lugar a los artefactos que se relacionan:

Rol	Actividad	Artefacto	Tipo de Artefacto
Analista del sistema	Desarrollar Documento Visión.	Requisitos de los Stakeholders	Entrada
	Gestionar Dependencias.	Plan de Gestión de Requisitos	Entrada y Salida
		Documento de Cambios en los Requisitos	Entrada y Salida
	Encontrar actores y casos de uso	Modelo de Caso de Uso	Entrada y Salida
		Caso de Uso	Salida

Tabla 10: Relación rol, actividad y artefacto en Definir Limites del Sistema.

Las actividades que se enfocan en el análisis del problema y la comprensión de las necesidades de los stakeholders crean iteraciones tempranas de definiciones del sistema, incluso un primer contorno de los requisitos detallados. Estas definiciones tienen lugar durante las fases de Inicio y Elaboración, sin embargo, se pueden volver a examinar, según sea necesario, en la gestión de alcance respondiendo a los cambios de requisitos y otros asociados a las condiciones del proyecto. Definiendo el sistema se identifican los actores y el Caso de Uso Detallados. Igualmente extiende los requisitos no funcionales globales como están definidos en las Especificaciones de Requisitos del Software.

2.5.4. Administrar el alcance del sistema.

El alcance de un proyecto está definido por el juego de requisitos asignado a él. Administrar el alcance es una actividad continua que requiere desarrollo iterativo o incremental e irrumpe el alcance del proyecto en pedazos menores más manejables. El propósito es desarrollar el alcance del sistema tan explícito como sea posible y enfocar, en un cuerpo manejable de trabajo, los requisitos para la iteración. El punto principal de esta actividad es la:

- ♣ Priorización de requerimientos con Casos de Uso.

El rol de analista del sistema es el responsable de determinar valores de prioridad, esfuerzo, costo y riesgo. Éstos se usarán por el personal en el flujo Gestión de Proyecto para idear cada iteración y permitirá al Arquitecto de Software identificar el escenario arquitectónicamente significativo o casos del uso completos ayudando a definir la vista del caso de uso de la arquitectura.

Las actividades que se desarrollan durante esta actividad y los artefactos que se generan en ella son:

Rol	Actividad	Artefacto	Tipo de Artefacto
Analista del sistema	Desarrollar Documento Visión	Requisitos de los Stakeholders	Entrada
	Gestionar Dependencias	Plan de Gestión de Requisitos	Entrada
		Documento de Cambios en los Requisitos	Entrada y Salida
Arquitecto de Software	Priorizar Caso de Uso	Modelo de Caso de Uso	Entrada
		Caso de Uso	Entrada
		Documento de la Arquitectura del Software	Salida

Tabla 11: Relación rol, actividad y artefacto en Administrar el Alcance del Sistema.

Un mejor entendimiento de la funcionalidad del sistema puede ser formulado al punto que la mayoría de los actores y casos del uso (por ejemplo el 80%) se ha identificado y perfilado. Los requisitos no funcionales que no encajan en el Modelo del Casos de Uso del Sistema o son generales para los casos del uso múltiples, deben documentarse en las Especificaciones de Requisitos del Software.
En el flujo de trabajo de Requerimiento una vez que se identifican los casos de uso para conformar la arquitectura candidata, se inicia el Documento de la Arquitectura del Software como uno de los artefactos que genera administrar el alcance del sistema, pues se cuenta con una de las vistas lógicas

para conformar este documento que se enriquece durante el desarrollo de los restantes flujos de trabajo del ciclo de vida del software. Se desarrolla con mayor importancia en el Análisis y Diseño donde se recogen todas las vistas arquitectónicas (Caso de Uso, Lógica, Proceso, Implementación y Despliegue).

2.5.5. Refinar la definición del sistema.

Refinar la definición del sistema lleva más allá de la comprensión del alcance del proyecto reflejada en el juego priorizado[15] que ofrece el producto. El rendimiento de esta actividad es más profundo entendiendo la funcionalidad del sistema expresado en los requisitos refinados en el artefacto Especificación de Requisitos de Software. El propósito de refinar la definición del sistema durante la captura de los requisitos es filtrarlos en orden para obtener un acuerdo que entienda de la definición del sistema. Se establecen como puntos principales:

- Especificar los requerimientos: Funcionales y no Funcionales.
- ¿Cómo detallar un Caso de Uso?
- Elaborar diagramas de Actividades.

El especificador de requisitos, para garantizar el desarrollo de este paso del procedimiento, realiza las siguientes actividades generando los artefactos que se asocian a continuación:

Rol	Actividad	Artefacto	Tipo de Artefacto
Especificador de Requisitos	Detallar Casos de Uso	Requisitos de los Stakeholders	Entrada
		Caso de Uso	Entrada
		Plan de Gestión de Requisitos	Entrada
	Detallar Requisitos del SW	Documento de Cambios en los Requisitos	Entrada y Salida
		Especificación de Requisitos del Software	Entrada y Salida

Tabla 12: Relación rol, actividad y artefacto en Refinar la Definición del Sistema.

El artefacto Especificación de Requisitos del Software puede tomar los formularios de Caso de uso Detallado y en algunos casos puede desarrollarse una especificación formal. Este trabajo empieza típicamente repasando las definiciones del actor existentes, entonces continúa con detallar los casos de uso que se han perfilado previamente para cada actor.

2.5.6. Administrar los cambios de requerimientos.

La gestión de cambios a los requisitos es un proceso que se desarrolla durante todo el ciclo de vida del software con el objetivo de evaluar el impacto de los cambios y de gestionar los cambios

aprobados. Es útil recordar la gran probabilidad de que cambien los requisitos durante el proceso de desarrollo y que pueden aparecer nuevos requisitos, aunque deben representar un por ciento mínimo del grupo total de requisitos identificados si la Definición del Sistema se desarrolló correctamente. Para ello se han definido como puntos principales:

- ♣ Cambios en los requerimientos.
- ♣ Métricas de Administración de Requerimiento.
- ♣ Trazabilidad e impacto de Requerimiento.

En Administrar cambios en los requisitos los roles analista del sistema y revisor técnico realizan las siguientes actividades dando lugar a los artefactos que aquí se relacionan:

Rol	Actividad	Artefacto	Tipo de Artefacto
Analista del Sistema	Estructurar Modelo de Caso de Uso	Modelo de Caso de Uso	Entrada y Salida
		Plan de Gestión de Requisitos	Entrada
	Gestionar Dependencias	Caso de Uso	Entrada y Salida
Revisor Técnico	Revisar Requisitos	Documento de Cambios en los Requisitos	Entrada y Salida

Tabla 13: Relación rol, actividad y artefacto en Administrar Cambios en los Requisitos.

El seguimiento de la historia de los requisitos se considera un paso importante durante la gestión de los mismos, para ello se utiliza el Documento de Cambios en los Requisitos. El propósito de esta plantilla es describir cómo el proyecto preparará y manejará los artefactos de requisito, el tipo de requisito asociado, y sus atributos. El plan también dirige cómo se manejará la trazabilidad. Para ello se realiza una matriz de seguimiento como forma de poseer la documentación y el control de todos los cambios realizados en el software.

2.6. *Métricas aplicables al procedimiento.*

2.6.1 Métricas de la calidad en el proceso de especificación de requisitos.

Para garantizar el buen desempeño de los proyectos se debe comenzar por la medición de la calidad de las Especificaciones de Requisitos. Las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o al menos interpretables, anotadas por importancia y estabilidad relativas, con su versión correspondiente, organizadas, con referencias cruzadas y especificadas al nivel correcto de detalle. Aunque muchas de las cualidades del software parecen ser de naturaleza

cualitativa Davis [16] sugiere que todas se puedan representar usando una o más métricas. Para obtener una medida del rendimiento se asume que hay n_r (requisitos en una especificación), tal como:

$$n_r = n_f + n_{nf}$$

Donde n_f es el número de requisitos funcionales y n_{nf} es el número de requisitos no funcionales. Para determinar la especificidad (ausencia de ambigüedad) de los requisitos existe una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

$$Q_1 = n_{ui} / n_r$$

Donde n_{ui} es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Cuanto más cerca de uno esté el valor de Q, menor será la ambigüedad de las especificaciones.

La complexión (constitución) de los requisitos funcionales puede determinarse calculando la relación:

$$Q_2 = u_n / (n_i \times n_s)$$

Donde u_n es el número de requisitos únicos de función, n_i es el número de entradas (estímulos) definidos o implicados por la especificación y n_s es el número de estados especificados, que no son más que:

- Requisitos propuestos.
- Requisitos aprobados.

La relación Q_2 mide el porcentaje de funciones necesarias que se han especificado para un sistema. Sin embargo, no trata los requisitos no funcionales. Para incorporarlos a una métrica global completa, debemos considerar el grado de validación de los requisitos.

$$Q_3 = n_c / (n_c + n_{nv})$$

Donde n_c es el número de requisitos que se han validado como correctos y n_{nv} el número de requisitos que no se han validado todavía

2.6.2 Métrica de calidad para la administración de cambio en los requisitos.

Esta actividad se propone para medir la calidad de una métrica cuyo objetivo es calcular las consecuencias que trae realizar grandes cambios en los requisitos en la medida que avanza el software, es decir en los restantes flujos de trabajo de cada fase del ciclo de vida de un software.

Previo al proceso de recolección de datos se llena el Documento de Cambios en los Requisitos. Se cuenta con la matriz de seguimiento de los requisitos para extraer los datos de la métrica, teniendo en cuenta los siguientes estados:

- Nuevo: Requisito que surge mediante un nuevo reajuste de la captura.
- Actualizado: Requisito que se ha modificado mediante un reajuste de la captura.
- Eliminado: Requisito que se ha desecho mediante un reajuste de la captura.
- Irrevocable: Requisito que no sufre cambios.

Las ecuaciones propuestas son las siguientes:

$$D = ((p_c * c_r) + r_i) / r_i$$

Donde D representa las consecuencias de que existan cambios en los requisitos, teniendo en cuenta que, mientras mayor sea D mayor serán las consecuencias. p_c es el peso de cambio en el flujo de trabajo que se realiza multiplicado por la fase en que se encuentra; r_i es la cantidad de requisitos irrevocables (sin cambios) y C_r es el coeficiente de requisitos que se calcula de la siguiente forma:

$$C_r = cr_n + cr_e + cr_a$$

Donde r_n, r_a y r_e son la cantidad de requisitos nuevos, actualizados y eliminados respectivamente, y C es el coeficiente del grado de influencia el cual se obtiene mediante la tabla 14 que contiene el grado de influencia dependiendo de la escala de los cambios en los requisitos y otorgándole un valor a cada grado, llamado coeficiente.

Grado de Influencia	Escala	Coeficiente
Bajo	0 - 3	1
Medio	3 - 6	2
Alto	> 6	3

Tabla 14: Escala de coeficiente de cambios en los requisitos.

Después de definir cual es el coeficiente de cada cambio en los requisitos, se calcula el peso de cambio de los requisitos, multiplicando el peso en cada flujo de trabajo por el peso en la fase donde se encuentra tal y como se muestra en la tabla 15.

Flujo de Trabajo	Fases	Inicio	Elaboración	Construcción	Transición
	Peso	**0.2**	**0.4**	**0.6**	**0.8**
Requerimiento	**0.1**	0.02	0.04	0.06	0.08
Análisis y Diseño	**0.2**	0.04	0.08	0.12	0.16
Implementación	**0.3**	0.06	0.12	0.18	0.24
Prueba	**0.4**	0.08	0.16	0.24	0.32
Despliegue	**0.5**	0.1	0.2	0.3	0.4

Tabla 15: Matriz de cálculo de pesos.

Esta métrica procura medir la afectación que nos produce realizar un cambio en el ciclo de vida del software. En muchas ocasiones estos cambios no se controlan ni se documentan provocando la desorganización y el descontrol del producto a realizar. Por tal motivo contiene coeficientes de cambios, para asegurarse de que mientras más avanzado esté el software, mayores serán las consecuencias de la realización de cambios desmedidos.

2.7. Herramienta propuesta para llevar a cabo la trazabilidad de los requisitos.

Los requisitos constituyen uno de los primeros entregables, si no el primero, en el proceso de desarrollo de aplicaciones software. De hecho, es la aparición de unas determinadas necesidades lo que motiva el desarrollo de un sistema o aplicación, el cual tiene como fin solucionar esas necesidades o deficiencias detectadas. Los requisitos son estas necesidades plasmadas de forma clara y concisa. Resulta por tanto evidente que esta parte de Gestión de Requisitos, es una de las más importantes del proceso de desarrollo y que en mayor o menor medida condicionará el éxito del proceso global.

El proceso de GR, no se basa únicamente en la educción de los mismos, si no que implica más tareas, por lo que surge la necesidad de ayudarse de herramientas que apoyen o den soporte a todas estas tareas, entre las cuales podemos destacar:

- Recogida de requisitos.
- Formalización de requisitos.
- Revisión y Gestión.

Las herramientas que dan soporte a estas tareas son las conocidas como Herramientas de GR.

2.7.1 Open Source Requirement Management Tool (OSRMT)

Es una herramienta de software libre, llevada a cabo por un único desarrollador y escrita en lenguaje Java, motivo por el cual puede ser usada en cualquier sistema operativo, es decir, funciona bajo Windows o Linux, principal motivo de su elección.

Se trata de una herramienta de Requisitos, que permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos.

Características y funcionalidad básica.

Los diversos módulos integrados en la herramienta son:

- Administración y Configuración.
- Gestión de documentos de ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba).
- Trazabilidad entre documentos de trabajo.
- Informes y estadísticas.

Figura 11: Herramienta Open Source Requirement Management Tool.

Las funcionalidades ofrecidas por la herramienta son:

1. Gestión de requisitos, diferenciando entre:
   - Requisitos.
   - Funcionalidades.

- ♣ Requisitos técnicos.
- ♣ Casos de prueba.

2. Trazabilidad entre todos los documentos de trabajo:
   - ♣ Requisito-Requisito (control de versiones y dependencia entre requisitos).
   - ♣ Requisito-Funcionalidad.
   - ♣ Requisito-Caso de Prueba.
   - ♣ Visualización de la matriz de trazabilidad.
   - ♣ Árbol de trazabilidad para facilitar las auditorias.
   - ♣ Gráfico de dependencias entre documentos de trabajo para poder determinar el impacto de un cambio.
3. Personalización y configuración:
   - ♣ Definición de los atributos de una funcionalidad.
   - ♣ Definición de los atributos de un requisito.
   - ♣ Definición de los atributos de un caso de prueba.
   - ♣ Valores predefinidos para cada usuario (prioridades por defecto y estado por defecto).
   - ♣ Personalización de vistas.
4. Representación jerárquica de los documentos de trabajo.
5. Definición de casos de prueba mediante pruebas para cada uno de los pasos del caso de uso.
6. Descripción de un requisito mediante la secuencia de pasos de su caso de uso.
7. Gestión de la configuración:
   - ♣ Almacenamiento de quién y cuándo cambia qué.
   - ♣ Versionado de los documentos de trabajo.
8. Posibilidad de almacenamiento de ficheros binarios adjuntos o hipervínculos.
9. Gestión de usuarios.
10. Acceso restringido a usuarios registrados:
    - ♣ Gestión de privilegios para determinadas tareas.
11. Búsquedas avanzadas (filtros, órdenes) sobre los documentos de trabajo registrados.
12. Informes y estadísticos:
    - ♣ Básicos.
    - ♣ Específicos creados por el usuario.
    - ♣ A partir de los resultados de búsquedas avanzadas.
    - ♣ Exportados a HTML o PDF.

13. Herramientas de migración para los diversos cambios de versiones.
14. Múltiples idiomas (importación y exportación para dar soporte a diversos idiomas).

Ventajas e inconvenientes.

La herramienta presenta las siguientes ventajas:

- La visualización de requisitos en forma jerárquica es intuitiva y fácil de manejar.
- Su licencia es GPL (Licencia Pública General).
- Es un desarrollo basado en Java, por lo que es multiplataforma.
- Las nuevas versiones incorporan un cliente Web para permitir accesos desde internet.
- Como herramienta open source de GR no tiene mucha competencia en cuanto a la funcionalidad ofrecida.
- Tiene una buena documentación pese a tratarse de una herramienta muy reciente.
- A pesar de ser llevada a cabo por un único desarrollador, el ritmo de mejoras y nuevas versiones es constante a lo largo del último año.
- Existen muchas opciones para configurar y personalizar la herramienta a las necesidades concretas de una organización.
- Lleva incorporado un sistema de gestión de la configuración que permite definir líneas base.
- Existe un gran soporte para mantener la trazabilidad entre los documentos.

Los principales inconvenientes que se han observado son los siguientes:

- Es un desarrollo llevado a cabo por una persona individual, por lo que existe el riesgo de que no sea sostenible a lo largo del tiempo.
- No existe un soporte empresarial.
- Las nuevas versiones no están planificadas ni se anuncian claramente las mejoras que serán incorporadas. Es posible que las nuevas versiones no sean compatibles con las anteriores.
- No es posible generar automáticamente un documento de requisitos para entregar al cliente.
- Algunas funcionalidades no han sido desarrolladas completamente y están a medias.
- La interfaz de usuario es en ocasiones lenta.
- Se ofrecen pocos mensajes de confirmación y aviso al usuario.

Valoración general

Dentro de las herramientas open source que abarcan la ingeniería de requisitos, se trata sin duda de una de las que mejores funcionalidades. Con muchas opciones de configuración, que permite personalizar la herramienta a las necesidades concretas de una organización, esta aplicación cubre los

principales aspectos relacionados con la GR: su registro, definición, categorización, seguimiento y trazabilidad con el resto de documentos de trabajo.

2.8. Conclusiones.

Los proyectos productivos asociados al Centro UCID necesitan mejorar de forma continua la manera de gestionar los requisitos como garantía a la producción de productos y servicios de mayor calidad. La excelencia en la Gestión de Requisitos se ha convertido en una ventaja competitiva de primer orden.

Refinar los requisitos y controlar sus cambios supone:

- Analizar la línea base.
- Realizar el canal y control de cambios.
- Efectuar la ejecución de los cambios.
- Mantener la trazabilidad.
- Controlar las versiones.

En el presente capítulo se describió el procedimiento propuesto para desarrollar la GR aplicando métricas de calidad. Para el futuro desarrollo del mismo se recomienda el uso de OSRMT como herramienta de apoyo que viabilice la medición de la trazabilidad de los requisitos. Para el perfeccionamiento de este procedimiento se aplicó una lista de auto-chequeo dirigido a los desarrolladores del MINFAR y se utilizaron un grupo de plantillas.

La propuesta realizada en este capítulo facilita el modelado, despliegue y mejora continua de los requisitos. Aplicarla nos traería grandes ventajas:

- Minimiza el desarrollo engorroso del flujo de trabajo Requerimiento generando los artefactos necesarios de cada actividad.
- Facilita técnicas para mejorar la captura de requisitos.
- Ayuda a mejorar la calidad del software.
- Mantiene la trazabilidad de los requisitos con una matriz de seguimiento.
- Guía el software por el camino correcto, permitiendo un entendimiento entre desarrolladores y cliente.
- Garantiza “medir” la calidad del software mediante métricas.
- Explica con claridad y exactitud las actividades para obtener una buena captura de requisitos.
- Posibilita obtener requisitos sin ambigüedades y con un mayor nivel de funcionalidad.

CAPÍTULO III: VALIDACIÓN DEL PROCEDIMIENTO.

3.1. Introducción.

El desarrollo de un procedimiento que responda a la Gestión de los Requisitos incluyendo la aplicación de métricas de calidad, lista de auto-chequeo y técnicas para la captura de requisitos proporciona una mejora basada en la reducción de los artefactos y actividades para hacer más fácil el flujo de Requerimiento.

Para la implementación de la propuesta es preciso realizar la validación de la misma. Los datos de este proceso fueron recogidos a través del panel de expertos y ayudan a conocer la utilidad del procedimiento. La técnica aplicada recibe el nombre de Método Delphi [22].

3.2. El método Delphi para la validación del procedimiento.

Métodos de expertos: Se basan en la consulta a personas que tienen grandes conocimientos sobre el entorno en el que la organización desarrolla su labor. Estas personas exponen sus ideas y finalmente se redacta un informe en el que se indican cuáles son, en su opinión, las posibles alternativas que se tendrán en el futuro.

Los métodos de expertos utilizan como fuente de información un grupo de personas a las que se supone un conocimiento elevado de la materia que se va a tratar. Estos métodos se emplean cuando se da alguna de las siguientes condiciones:

- No existen datos históricos con los que trabajar.
- El impacto de los factores externos tiene más influencia en la evolución que el de los internos.
- Las consideraciones éticas o morales dominan sobre las económicas y tecnológicas en un proceso evolutivo.

El nombre de este método de prospección proviene del oráculo de Delphos, que se encontraba en la antigua Grecia, al que se acudía para hacer preguntas al Dios a través de una sacerdotisa. A pesar del carácter siempre ambiguo de las respuestas, el oráculo de Delphos era el que tenía mejor reputación por la certeza de sus predicciones.

Delphi es uno de los métodos de pronosticación más confiables, constituye un procedimiento para confeccionar un cuadro de la evolución de situaciones complejas a través de la elaboración estadística de las opiniones de un grupo de expertos en el tema tratado.

Este método presenta tres características fundamentales, que le permiten garantizar la calidad de los resultados, para lanzar y analizar la Delphi:

1. Anonimato.
2. Iteración y realimentación controlada.
3. Respuesta del grupo en forma estadística.

Para la aplicación práctica del método es necesario considerar:

- ♣ Selección de los expertos.
- ♣ Cálculo del coeficiente de conocimiento.
- ♣ Elaboración del cuestionario.
- ♣ Análisis de los resultados de la validación del procedimiento.

3.3. Selección de los expertos.

Seleccionar el panel de expertos y conseguir su compromiso de colaboración, con independencia de sus títulos, su función o su nivel jerárquico. El experto será elegido por su capacidad de encarar el futuro y posea conocimientos sobre el tema consultado. Las personas que sean seleccionadas no sólo deben ser grandes conocedores del tema sobre el que se realiza el estudio sino deben presentar una pluralidad en sus planteamientos.

La selección del posible equipo de experto se realizó bajo los siguientes criterios:

- ♣ Graduado de nivel superior.
- ♣ Vinculación al desarrollo de productos informáticos.
- ♣ Conocimientos y habilidades en actividades de desarrollo de software.
- ♣ Conocimiento en la materia de Gestión de Requisitos y métricas de calidad.

A continuación se confeccionó un listado inicial de expertos teniendo en cuenta su experiencia en las áreas identificadas. El listado estuvo conformado por 15 expertos: 5 líderes de proyectos, 1 ingeniero principal, 9 miembros del Consejo de Calidad de la Universidad, de ellos 9 han realizado estudios sobre el tema.

Luego de la solicitud del consentimiento de los posibles involucrados para participar en la validación del procedimiento, 8 estuvieron de acuerdo.

De manera particular, las características de los expertos, influyen en los resultados obtenidos y dan la medida del grado de confiabilidad del mismo. En el Anexo 9 aparece una caracterización de ellos.

3.4. Cálculo del coeficiente de competencia.

Para la selección de los expertos es útil emplear la valoración por competencias. Este método consiste en calcular el coeficiente de competencia (k) del experto a partir de la autovaloración del experto sobre su conocimiento o información sobre el tema (kc) y el coeficiente de argumentación o valoración (ka) mediante la siguiente ecuación:

$$k = (kc + ka) / 2$$

El cálculo del coeficiente de conocimiento se obtiene de la primera pregunta del cuestionario de coeficiente de conocimientos de los expertos (Ver Anexo 10), multiplicando el número marcado por 0.1.

Para el coeficiente de argumentación, las marcas de los expertos se traducen a puntos, según la siguiente escala:

No.	Fuentes de argumentación	Grado de influencia		
		Alto	Medio	Bajo
1.-	Análisis realizado por Ud.	0.3	0.2	0.1
2.-	Experiencia.	0.5	0.4	0.2
3.-	Trabajos de autores nacionales.	0.05	0.05	0.05
4.-	Trabajos de autores extranjeros.	0.05	0.05	0.05
5.-	Su propio conocimiento del tema.	0.05	0.05	0.05
6.-	Su intuición.	0.05	0.05	0.05

Tabla 16: Grado de conocimiento.

En la tabla anterior el valor de las casillas marcadas por el experto se sumará dando el valor de ka. Después de obtener los dos valores de la ecuación se suma y divide por dos.

Para saber el código de interpretación de los coeficientes de competencias se analiza lo siguiente:

- Si 0,8 < k < 1,0 coeficiente de competencia alto.
- Si 0,5 < k < 0,8 coeficiente de competencia medio
- Si k < 0,5 coeficiente de competencia bajo.

Aquellos expertos que se encuentre en nivel bajo no pertenecerán al panel. Los expertos escogidos poseen coeficiente de competencia altos y medios.

3.5. Elaboración del cuestionario.

Conociendo que la calidad de los resultados depende, sobre todo, del cuidado que se ponga en la elaboración del cuestionario y en la elección de los expertos consultados. Las preguntas, en su conjunto, se elaborarán previendo que sus resultados permitan prever las transformaciones más importantes que puedan producirse luego de aplicado el procedimiento en los proyectos productivos de la UCID.

Luego de haber seleccionado los expertos calculado su coeficiente de conocimiento se elabora un cuestionario para validar la propuesta del procedimiento. La encuesta se lleva a cabo de una manera anónima (se realiza haciendo uso del correo electrónico) para evitar influenciar en la opinión de los expertos.

Las preguntas se refieren a las probabilidades de acontecimientos con relación al tema de estudio. Los cuestionarios se elaboraron de manera que faciliten, en la medida que la investigación lo permita, la respuesta por parte de los consultados.

En ocasiones, se recurre a respuestas categorizadas (ejemplo: Si/No; Muy Adecuado/Adecuado/Poco Adecuado/No Adecuado, Innecesarias/Necesarias y suficientes/Necesarias y no suficientes) y después se tratan las respuestas en términos porcentuales.

La calidad de los resultados depende, sobre todo, del cuidado que se ponga en la elaboración del cuestionario y en la elección de los expertos consultados.

3.6. Análisis de los resultados de la validación del procedimiento.

Con el propósito de validar el procedimiento propuesto, se aplicó al panel de experto una encuesta con el objetivo de conocer si la propuesta es efectiva para darle solución al problema existente en la UCID, comprobar que no posea ambigüedades, que sea correcta, completa, fácil de realizar y precisa. Además conocer de las recomendaciones que los expertos puedan aportar (Ver Anexo 11).

Los resultados arrojados por la encuesta fueron los siguientes:
En el gráfico que ponemos a continuación se hace referencia a las 3 primeras preguntas del cuestionario en las cuales se aborda sobre:

1. La importancia de la realización de un procedimiento para la GR.
2. La necesidad que el centro UCID adecue el procedimiento definido.
3. La efectividad de la aplicación del procedimiento propuesto.

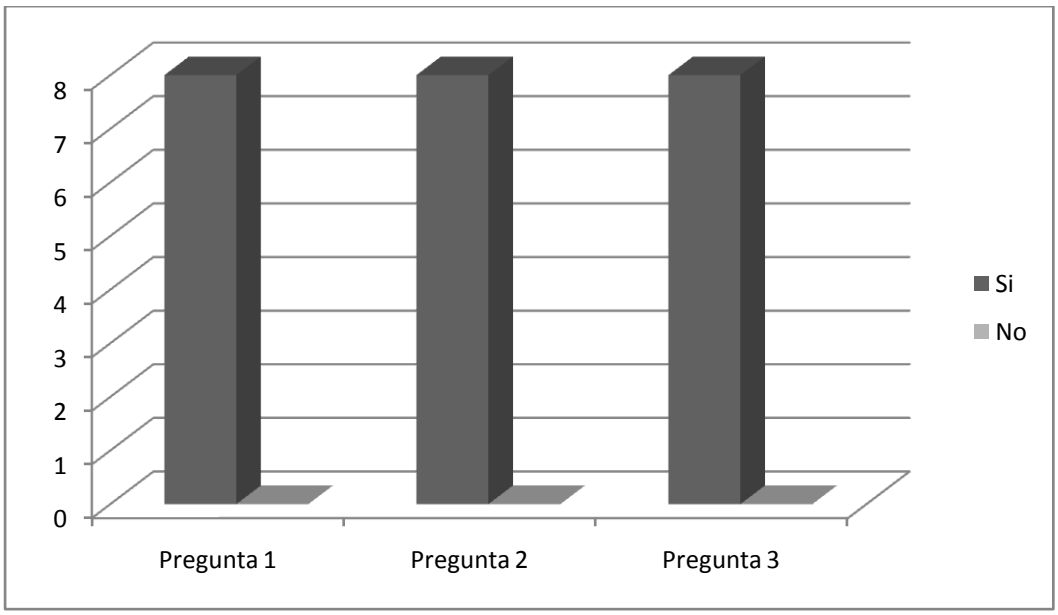


Figura 12: Utilidad del procedimiento.

Necesidad, corrección y completitud del procedimiento.

Para valorar la necesidad del procedimiento se elaboró la pregunta 4 con los incisos (a, b, c) que evalúan el desarrollo de las actividades, los artefactos relacionados con ellas y el uso de una matriz de seguimiento para colaborar con la trazabilidad de los requisitos. El resultado se muestra en la siguiente tabla:

Incisos	Cantidad de expertos de acuerdo con el criterio		
	Innecesarias	Necesarias pero no suficientes	Necesarias y suficientes
a. Las actividades propuestas son:	0	2 (25%)	6 (75%)
b. Los artefactos propuestos son:	0	3 (37.5%)	5 (63.5%)
c. Para seguir la traza de los requisitos la matriz de seguimiento definida en la plantilla de Gestión de Cambios en los requisitos es:	0	2 (25%)	6 (75%)

Tabla 17: Resultado de la utilidad del procedimiento.

Los incisos d, e y f de la propia pregunta se refieren a la corrección y completitud de la propuesta y los resultados se encuentran graficados en la siguiente figura:

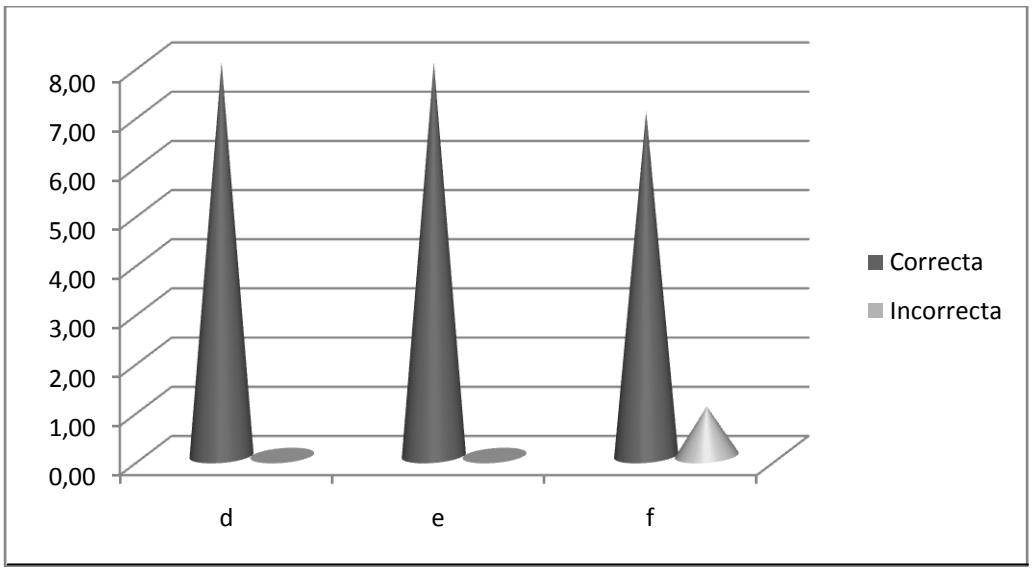


Figura 13: Corrección y completitud de la propuesta.

En la pregunta 5 se pone a consideración de los expertos si la aplicación de métricas en la propuesta realizada constituye un paso de avance para el proceso de desarrollo del software. El resultado se muestra en el siguiente gráfico:

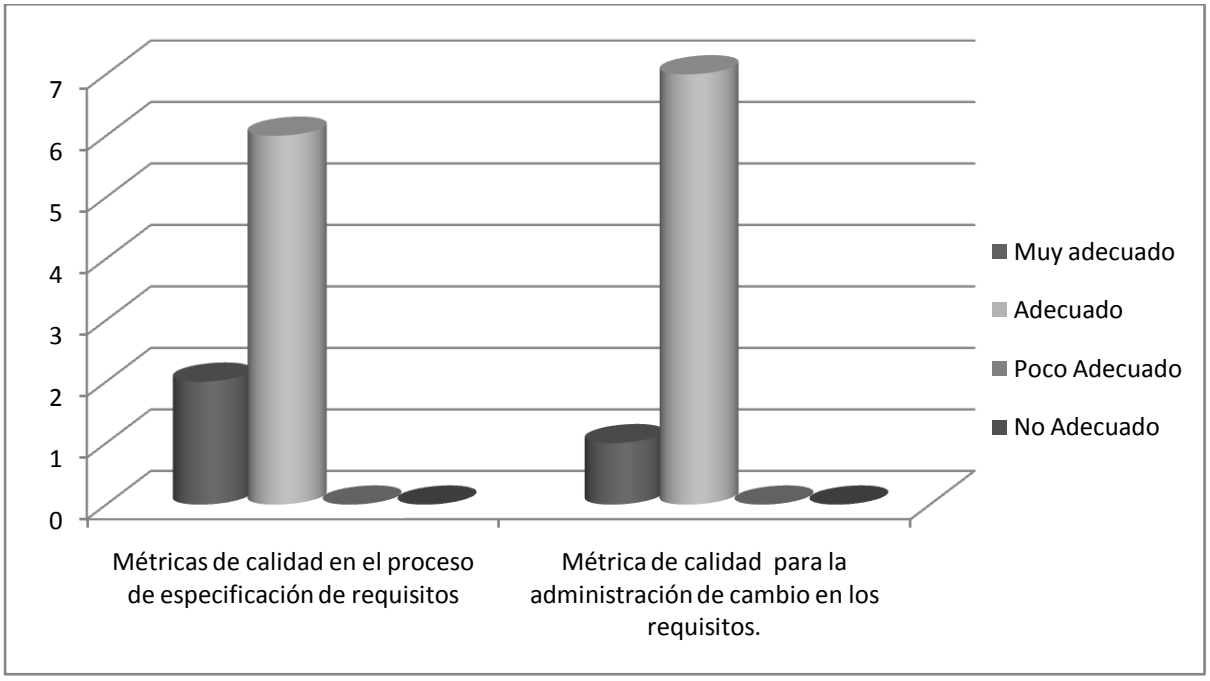


Figura 14: Resultado de la aplicación de métricas.

Los datos para el cálculo de las métricas fueron evaluados de la siguiente forma.

Métricas	Cantidad de expertos de acuerdo con el criterio		
	Innecesarios	Necesarios pero no suficientes	Necesarios y suficientes
Métricas de calidad en el proceso de especificación de requisitos	0	2	6
Métrica de calidad para la administración de cambio en los requisitos.	0	1	7

Tabla 18: Resultado de la necesidad de los datos en las métricas.

En la pregunta 6 se recogen otros criterios emitidos por los expertos y que ayudan a la mejora del procedimiento propuesto. A continuación se relacionan esas opiniones:

- La implantación del procedimiento puede resultar muy útil.
- Logrará estandarizar la forma de capturar requisitos en la UCID.
- Propondrá una forma de gestionar los cambios en los requisitos que hasta el momento no se realiza.
- Propone métricas para medir el avance del procedimiento.
- Es un procedimiento que es fácil de realizar, sin muchos artefactos y se encuentra escrito de forma clara para su implantación.
- Aportará elementos que sirvan para desarrollar con calidad la Gestión de Requisitos en el centro UCID.

Recomendaciones:

- Podría ser positivo implementar la captura del lenguaje común. No es crear todo un artefacto y una reunión para ello, sino que el ingeniero de requisitos a partir de las diferentes conversaciones con los clientes debe ir determinando aquellas expresiones o palabras que puedan resultar ambiguas y susceptibles a malas interpretaciones. O sea, no es necesario definir un proceso para ello, sino irlo realizando de forma intuitiva y por supuesto, luego registrarlo.
- En la captura de requisitos se pudiera agregar que las entrevistas se deben grabar para revisarlas posteriormente y poder aclarar dudas con respecto a las mismas.

3.7. Conclusiones.

Delphi es sin duda una técnica que desde hace unos cuarenta años ha sido objeto de múltiples aplicaciones en el mundo entero. A partir del procedimiento original, se han desarrollado otras aproximaciones.

El análisis de los resultados anteriores permite afirmar que de manera general el procedimiento fue evaluado por los expertos como válido, correcto, completo y seguro para desarrollar la Gestión de Requisitos en los proyectos productivos de la Unidad de Compatibilización, Integración y Desarrollo de software para la defensa, aplicándoles métricas de calidad y además con la propuesta de una herramienta para la trazabilidad de los requisitos. Es de vital importancia que se aplique el procedimiento propuesto sin violar sus pasos originales para lograr un software de alta calidad.

CONCLUSIONES

Al finalizar la presente investigación se logra el cumplimiento de los objetivos trazados y se concluyó lo siguiente:

- Se logró integrar los conocimientos adquiridos a lo largo de la carrera para la realización de este trabajo.
- Se demostró en esta investigación que para minimizar la complejidad y relatividad inherentes a la GR en el software, se hace necesaria la aplicación de un procedimiento que permita mantener la calidad de productos software.
- Se demostró que no solo es posible “medir” la calidad, sino también “construir” la calidad durante el proceso de desarrollo, utilizando para ello las actividades de la GR.
- Se diseñaron plantillas de GR que ayudan a mejorar este proceso.
- Se hizo una propuesta de métricas incluidas en el procedimiento de GR, que permiten hacer un análisis de la toma de medidas oportunas durante la ejecución del proceso de GR.
- Se propicia el mejoramiento continuo del proceso al poder contar con un conjunto de métricas que permitan medir la efectividad del proceso de GR.
- El procedimiento abarca actividades, artefactos y roles necesarios para darle un control y seguimiento a la GR.

RECOMENDACIONES

Para extender la investigación presentada en este trabajo de diploma se recomienda:

- Se aplique el procedimiento en los proyectos asociados al Centro UCID, para obtener beneficios del mismo.
- Aplicar las métricas propuestas para la medición de la calidad en la GR.
- Aplicar la matriz de seguimiento para la trazabilidad de los requisitos en ciclo de vida del software.
- Identificar otras métricas que se pueden emplear en aras de medir la calidad en la GR, basándose en la clasificación de características de calidad establecidas.

REFERENCIAS BIBLIOGRÁFICAS

1. Pressman, R.S., *Ingenería de Software. Un enfoque práctico*. 1998.
2. *ISO 9001:2000*. [cited; Available from: http://www.iso.com/.
3. (2004) *Informe de investigación del SEI (Software Engineering Institute - Instituto de Ingeniería del Software)*.
4. 8402, I., *Calidad. Definiciones*, in *La revista del empresario cubano*. 1995.
5. IEEE, S.-. *Calidad de Software.* Marzo 2006.
6. Kent, K.M.a.J., *Interpreting the CMMI: A Process Improvement Approach*. 2003.
7. Letelier, E.A.S.a.E.A.P., *Mejorando la gestión de historias de usuarios en eXtreme Programing.*
8. Reifer, D.J., *Requirements Engineering* 1994. p. 1043-1054.
9. STD-610, I., ed. *IEEE Computer Society*. 1990.
10. Sons, P.S.J.W., *Requirement Engineering: a good practice guide.* 1997, Ian Sommerville.
11. Gause, D.C., y G.M. Weiberg, *Exploring Requirements : Quality Before Desing*. 1989: Dorset House.
12. Zultner, R., *Building Software in Group*. 1990: American Programmer.
13. Bossrt, J.L., *Quality Function Deploymente: ASQC Press*. 1991.
14. M. Edwards , S.H., *A methodology for system requirements specification and traceability for large real- time complex systems.* 1991.
15. Palmer, J.D., *Traceability in Software Requirements Engineering, R.H. Thayer and M. Dorfman (Eds)*. 1997.
16. Davis, A., *Identifying and Measuring Quality in a Software Requirements Specification*. Mayo 1993.
17. Davis, A., *201 Principles of Software Development*. 1995, McGraw-Hill.
18. P.Sawyer, G.K., *SWEBOK: Software Requeriments Engineering Knogledge Area Description*. 1999.
19. Durán, A., *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. 2000, Universidad de Sevilla.
20. M. Báez, S.B., *Metodología DoRCU para la Ingeniería de Requisitos" WER´2001.* . 2001, Buenos Aires, Argentina.
21. Pohl, K., *Requeriments Engineering: An Overview*. 1997.
22. *Método Delphi*. 2007 [cited; Available from: www.gtic.ssr.upm.es/encuestas/delphi.htm.

BIBLIOGRAFÍA

1. Aguillón, E.R., *Métricas para la Gestión de Proyectos de Software*. 2007, Instituto Tecnológico de Morelia.
2. [cited 2007 Diciembre 11]; Available from: http://www.metricas.com/.
3. Engineers, I.o.E.a.E. *Guide to the Software Engineering Body of Knowledge*. 2001 [cited 2008 Marzo 14]; Available from: http://www.swebok.org.
4. *Extreme Programming. A gentle introduction.* 1999 [cited 2008 Mayo 20]; Available from: http://www.extremeprogramming.org/
5. *"Estudio de herramientas de certificación de madurez de procesos de desarrollo".* 2006.
6. Fuentes, J.M., *Optimización del proceso de Gestión de Requisitos en el desarrollo de aplicaciones software*.
7. Gracia, J. (2003) *Prácticas y métodos para mejorar el desarrollo de Proyectos de Software*. **Volume**,
8. Henderson, S., *B. Object-Oriented Metrics*. 1993: Prentice Hall.
9. I. Jacobson, G.B.a.J.R., ed. *El Proceso Unificado del Desarrollo de Software*. ed. E. Wesley. 2000: Madrid.
10. Ince, D., *An Introduction to Software Quality Assurance and its Implementation*. 1994, McGraw-Hill.
11. Monzón, J.L.F.A., *A Metamodel and a Tool for Software Requirements Management* 2000, Berlin, Germany.
12. Myers, G., *El arte de probar software*. 1977, Kapelusz.
13. Pleeeger, N.F.a., *S. Software Metrics*. 1997, PWS.
14. Pressman, R.S., ed. *Ingeniería de Software, un enfoque práctico*. Quinta edición ed. 2002, McGraw-Hill.
15. Rolando Armas Andrade, A.C.G., Maite Montes Beobide, José Antonio and Gutierrez de Mesa, *Desde ISO 9001 Hacia CMMI, Pasos para la mejora de los procesos y métricas*. Enero, 20072008 [cited.
16. 2008 Mayo 5]; Available from: http://www.dsdm.org/
17. 2008. *Gestión de Calidad*. [cited 2008 Febrero 20]; Available from: http://www.calidaddelsoftware.com.
18. Ullman, R.D.a.R., *Quality Assurance for Computer Software*. 1982, McGraw-Hill.

ANEXOS

Anexo 1: Encuesta para recoger las experiencias de los proyectos desarrollados en el centro UCID.

Encuesta para Recoger las experiencias de los Proyectos desarrollados en el centro Unidad de Compatibilización Integración y Desarrollo.

Nombre del Proyecto: _____ Proyecto Nacional ___

Nombre del Líder de Proyecto: _____

Categoría de Proyecto:

___Portal ___Multimedia

___Gestión ___Software Base

Otros_____

Metodología de desarrollo de software

¿Utilizó Metodología de desarrollo?

__Sí __No

Si lo utilizó diga el tipo de metodología

__Metodología Ágil. __ Metodología Robusta.

¿Cuál metodología utilizó?

Metodología Ágil.

____Programación Extrema (XP)

____SCRUM

____Metodología de Administración de Relaciones (RMM).

Otras_____

Metodología Robusta.

___ Proceso Unificado de Rational (RUP)

___Microsoft Solution Framework (MSF)

Otras _____

Procedimiento de Gestión de Requerimiento (GR)

¿Realizó un procedimiento para la GR?

__Sí __No

¿Del procedimiento diga cuáles de estas actividades se realizan?

___ Análisis del problema.

___ Entender las necesidades de los stakeholders.

___ Definir el sistema.

___ Administrar el alcance del sistema.

___ Refinar la definición de sistema.

___ Administrar los cambios de requerimientos.

Otras_____

¿Después de la captura de requisitos se realizó algún cambio?

__Sí __No

¿En qué Flujo de Trabajo se realizó?

__Requerimiento __Despliegue

__Análisis y Diseño __Configuración de Cambio

__Implementación __Administración de Proyecto

__Prueba __Ambiente

¿Cuáles fueron las causas?

__Petición de cambios en los requisitos. __Insatisfacción con el software.

__No se entrevisto a la persona adecuada __Incorrecta comunicación con el cliente

Otras_____

¿Se realizó la documentación de los cambios?

__Sí __No

En caso de que la respuesta sea negativa. ¿Por qué?

_____

_____.

Anexo 2: Plantilla Plan de Gestión de Requisitos.

Plan de Gestión de Requisitos

<Nombre del Proyecto>

<Nombre del producto>

<Versión>

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Reglas de Seguridad

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.Estas reglas son aplicables a las __ páginas de este documento.

1. Introducción

1.1. Alcance.

[Proyectos con los que se involucra el Plan]

1.2. Definiciones, Acrónimos y Abreviaturas.

1.3. Referencias.

[Lista de documentos a los que se hace referencia en el Plan]

Código	Título
[1]	Documento 1
[4]	Documento 2
[3]	Modelo de Diseño - Módulo de Administración v0.0

2. Gestión de Requisitos

2.1. Organización, responsabilidades e interfaces

[Describe quien es responsable del desarrollo de las actividades descritas en el flujo de trabajo de requisito]

2.2. Herramientas, ambientes e infraestructura

[Describe las herramientas de software que serán utilizadas para la gestión de los requisitos. Describe las herramientas y procedimientos usados para el control de versiones de los Requisitos generados durante todo el ciclo de vida]

3. Programa de Gestión de Requisitos

3.1 Identificación de Requisitos

[Describe como los elementos serán nombrados, marcados y numerados. Puede utilizarse la siguiente tabla]

Artefacto (Tipo de doc.)	Elemento de trazabilidad	Descripción
Solicitud de los involucrados (STR)	Solicitud de los involucrados (STRQ)	Principales solicitudes de los involucrados
Glosario (GLS)	Términos (TERM)	Términos necesarios para comprender el modelo
Visión (VIS)	Necesidades de los involucrados(NEC)	Principales necesidades de los involucrados y usuarios
Visión (VIS)	Características Básicas (CAB)	Condiciones o capacidades de esta libración del sistema

Modelo de Casos de Uso	Caso de Uso (CU)	Caso de Uso para esta versión del documento
Especificaciones Adicionales (EA)	Requisitos Adicionales (RA)	Requerimientos no funcionales que no fueron capturados en el modelo de CU

3.2 Seguimiento

3.2.1 Criterio de <elemento de seguimiento>

[Para cada elemento se definen un grupo de reglas o guías para aplicar la traceabilidad. Por ejemplo: Toda característica básica aprobada se le asocia uno o más casos de uso o uno o más requisitos complementarios.]

3.3 Atributos

3.3.1 Atributo para <elemento de seguimiento>

[Para cada elemento se identifican la lista de los atributos que serán utilizados para caracterizarlo. Algunos de los atributos pueden ser los siguientes:]

Estado

[Conjunto de posibles estado de un elemento.]

Propuesto	*[Se usa para describir características que están en discusión pero que aún no han sido revisadas y aceptadas]*
Aprobado	*[Característica que ya fue aprobada para su implementación]*
Rechazada	*[Característica rechazada]*
Incorporada	*[Característica incorporada en un momento específico del proyecto]*

Beneficio

[Que beneficio reporta al negocio]

Crítico	*[Es una característica esencial para el negocio]*
Importante	*[Es una característica importante que no incluirla pudiera afectar la satisfacción de los clientes y usuarios]*
Útil	*[No es una característica que afecte en la satisfacción del cliente]*

Esfuerzo

[Los requisitos son diferentes en cuanto al tiempo de desarrollo y otros recursos.]

Bajo	*[< 1 día]*

Medio	*[1 a 20 días]*
Alto	*[>20 días]*

Estabilidad

[% aceptable de cambios que pudiera tener el requisito.]

Bajo	*[< 10%]*
Medio	*[10-50%s]*
Alto	*[>80 %]*

4. Entregables del Proyecto

[Una lista tabular de los artefactos que serán creados durante el proyecto, incluso las fechas de entrega designadas.]

5. Gestión de Cambios a los requisitos

[Hacer referencia al documento de Cambios de requisitos]

Anexo 3: Plantilla de Requisitos de los Stakeholders.

Requisitos de los Stakeholders

<Nombre del Proyecto>

<Nombre del producto>

<Versión>

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

1. Introducción

1.1 Propósito

[Especificar propósito de la colección de peticiones de los involucrados]

1.2 Alcance

[Una pequeña descripción del alcance de la colección de las Peticiones de los Involucrados]

1.3 Definiciones, Acrónimos y Abreviaturas

[Proveer la definición de todos los términos, siglas y abreviaturas requeridas para poder interpretar apropiadamente los requisitos de los Stakeholder]

1.4 Referencias

[Lista completa de todos los documentos referenciados en el documento Requisitos de los Stakeholders]

1.5 Descripción General

[Esta subdivisión debe describir que es lo que contiene el resto del documento Requisitos de los Stakeholders así como también como es que están organizados.]

2. Establecer Stakeholders o Perfil de usuario

- Nombre: Empresa:
- Cargo:
- ¿Cuáles son sus responsabilidades importantes?
- ¿Qué documentos usted produce? ¿Para quién?
- ¿Qué problemas interfieren con su éxito?
- ¿Qué, en su caso, hace las tendencias de su trabajo más fácil o más difícil?

3. Evaluando el problema

- ¿Para qué tipo de problema, necesita bunas soluciones?
- ¿Cuáles son? *[Seguir Preguntando "Algo mas?"].*

Pregunta para cada problema:

- ¿Por qué existe este problema?
- ¿Cómo resolverlo?
- ¿Cómo te gustaría resolverlo?

4. Entendiendo el Ambiente del Usuario

- ¿Quiénes son los usuarios?
- ¿Cuál es su nivel educativo?
- ¿Tienen experiencias los usuarios con este tipo de aplicación?
- ¿Qué plataformas se encuentran en uso? ¿Cuáles son sus planes para las futuras plataformas?
- ¿Qué aplicaciones adicionales usted utiliza para nosotros poder interactuar con ella?
- ¿Cuáles son sus expectativas para la utilidad del producto?
- ¿Cuáles son sus expectativas para el tiempo de formación?
- ¿Qué tipo de documentos impresos y documentación en línea usted necesita?

5. Recapitulación de Entendimiento

- ¿Me puede decir todos sus problemas? *[Lista de los involucrados donde describen problemas de su propio mundo].*
- ¿Estos representan los problemas que está teniendo con la solución existente?
- ¿Cuáles otros problemas, en su caso, están experimentando?

6. Las aportaciones del analista a las partes interesadas del Problema (validar o invalidar las hipótesis)

- *[Si no se abordan]* ¿Con qué, en su caso, los problemas se asocian?: *[Lista de las necesidades o problemas adicionales usted piensa que debería afectar a los interesados o usuario]*

Pregunta para cada problema sugerido:

- ¿Se trata de un problema real?
- ¿Cuáles son las razones de este problema?
- ¿Cómo actualmente solucionar el problema?
- ¿Cómo le gustaría resolver el problema?
- ¿Cómo clasificar la solución de estos problemas en comparación a otros que has mencionado?

7. La evaluación de su solución (si procede).

- *[Resumir en una lista las principales capacidades de su solución propuesta]*
- ¿Cómo clasificar la importancia de estos?

8. Evaluación de la Oportunidad.

- ¿Quién necesita esta aplicación en su organización?
- ¿Cuántos tipos de usuarios podrían utilizar la aplicación?
- ¿Cómo valoraría usted una solución satisfactoria?

9. La evaluación de la fiabilidad, el rendimiento y las necesidades de apoyo

- ¿Cuáles son sus expectativas de fiabilidad?
- ¿Cuáles son sus expectativas para el rendimiento?
- ¿Apoyará usted el producto, o lo harán otros de apoyo?
- ¿Tiene usted necesidades especiales de apoyo? ¿Qué pasa con el mantenimiento y el servicio de acceso?

- ¿Cuáles son los requisitos de seguridad?
- ¿Cuáles son la instalación y configuración de los requisitos?
- ¿Cuáles son los requisitos de licencia especial?
- ¿Cómo el software será distribuido?

Otros Requerimientos

- En caso de cualquier reglamentación, requisitos ambientales o las normas, ¿cómo deben ser compatibles?
- ¿Puedes pensar en algún otro requisito que deberíamos conocer?

10. Preguntas de seguimiento.

- ¿Hay otras preguntas que quiera realizarme usted?
- Si tengo que continuar con el seguimiento de las preguntas, ¿puedo hacerle una llamada?
- ¿Estaría usted dispuesto a participar en una entrevista de requisitos?

11. Resumen del analista

- *[Resumir a continuación los 3 o 4 problemas de mayor prioridad para el usuario o involucrado]*

Anexo 4: Plantilla de Modelo de Caso de Uso del sistema.

Modelo de Casos de Uso del sistema

<Nombre del Proyecto>

<Nombre del producto>

<Versión>

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

1. Introducción

1.1 Propósito

[Resumen del propósito de este documento]

1.2 Alcance

[Breve descripción del alcance del modelo del negocio]

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

2. Breve Descripción del Sistema

[Enunciar el objetivo y describir de modo general las principales funcionalidades que tendrá el sistema, incluyendo los módulos (si ya están identificados)]

3. Actores del Sistema

[Se especifican todos los actores del negocio y se le asocia una descripción simple de cada uno]

Actor	Descripción

4. Diagrama de Casos de Uso del Sistema

[Figura que ilustre del modelo de casos de uso del sistema]

5. Especificación de los Casos de Uso

5.1 <Primer Caso de Uso del Sistema>

5.1.1 Descripción de Casos de Uso

Caso de Uso	*[Nombre del caso de uso]*
Actores	*[Actores, especificar quien inicia el CU]*
Propósito	*[Que función realiza el CU. De forma bien sintetizada, en una oración]*
Resumen	*[Breve descripción de lo que hace el CU debe quedar claro como se inicia y como termina y de que forma intervienen los actores en el CU]*
Precondiciones	*[Lo que se necesita para que se pueda ejecutar el CU]*
Postcondiciones	*[Lo que resulta del caso de uso. Especificar los posibles estados finales (como en el libro de RUP).Condiciones en las que queda el sistema cuando termina la ejecución del CU]*
Prioridad	
Responsabilidad	*[Responsabilidades del CU aquí se hace referencia a los requisitos funcionales con los que cumple este CU se escribe de la siguiente manera R y a continuación el número del requisito al que se hace referencia. Ejemplo R1, R1.1. Sucede que un CU satisface más de un requisito funcional]*

<table>
<tr><td>CU Relacionados</td><td colspan="2">[Nombre de los CU asociados con el tipo de asociación (include, extend, generalización...)]</td></tr>
<tr><td colspan="3">Descripción</td></tr>
<tr><td colspan="3">Interfaz</td></tr>
<tr><td colspan="3">[En cada lugar que se necesite antes del flujo de eventos al cual se corresponde, del caso de uso (si se necesita), y de cada una de las secciones (si estas existen y tienen interfaz). Graficar ventanas con sus componentes. La interfaz la enumeramos con número romano y con este la referenciamos donde quiera que haga falta dentro de la descripción del CU, le asociamos también un nombre que es el realmente aparecerá en la interfaz cuando se ejecuta el sistema.]</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>[Aquí se enumeran las acciones que se realizan de forma consecutiva para el flujo normal de eventos. La primera acción para el que la ejecute (ya sea actor o sistema) es como se inicia el CU.]</td><td colspan="2">[En la parte correspondiente a la Respuesta del Sistema debe ponerse en caso de ser necesario el estado en que se muestran los botones de las opciones que se ejecutan, si se muestran o no o si se muestran habilitadas o deshabilitadas.]</td></tr>
<tr><td colspan="3">Sección: Nombre de la sección (en caso de que exista)</td></tr>
<tr><td colspan="3">Interfaz (Relacionada con la sección si existe)</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>[Aquí se enumeran las acciones que se realizan de forma consecutiva para la sección se comienza por 1]</td><td colspan="2">[Aquí se enumeran las acciones que se realizan de forma consecutiva para la sección se comienza por 1]</td></tr>
<tr><td colspan="3">Cursos Alternos</td></tr>
<tr><td colspan="3">[Especificar la línea en que ocurre, tanto del flujo normal de eventos del caso de uso como de las secciones que tenga.]</td></tr>
</table>

[Si se decide tener un documento independiente para la definición de cada Caso de Uso, en esta sección se haría referencia a ese documento]

Anexo 5: Plantilla Caso de Uso detallado.

Plantilla Caso de Uso detallado

<Nombre del Proyecto>

<Nombre del producto>

<Versión>

Revisiones Históricas

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Descripción detallada de los casos de Uso

Caso de Uso	*[Nombre del caso de uso]*
Actores	*[Nombre de los actores que intervienen, si hay más de un actor se pone entre paréntesis quien es el que lo inicializa, de lo contrario no se especifica.]*
Propósito	*[Que función realiza el CU, de forma bien sintetizada, en una oración]*
Resumen	*[Breve descripción de lo que hace el CU debe quedar claro como se inicia y como termina y de que forma intervienen los actores en el CU]*
Responsabilidad	*[Responsabilidades del CU aquí se hace referencia a los requisitos funcionales con los que cumple este CU se escribe de la siguiente manera R y a continuación el número del requisito al que se hace referencia (Ejemplo R1ó R1.1). Si el caso de uso responde a mas de un requisito funcional se pone por ejemplo (R1-R6), de esta forma estaríamos diciendo que este caso de uso responde desde el requisito número 1 hasta el requisito número 6]*
CU asociados	*[Nombre de los CU asociados con el tipo de asociación (include, extend, generalización...), se pondría por ejemplo (El CU Buscar persona es una inclusión)]*
Precondiciones	*[Lo que se necesita para que se pueda ejecutar el CU]*
Requisitos especiales	*[No es obligatorio ponerlo, solo en los casos que lo lleve, precisar de qué manera, algunos de estos requisitos son de restricciones de tiempo de respuesta, seguridad, velocidad, disponibilidad, exactitud o uso de memoria afecta al CU]*
Descripción	
Interfaz (Se pone a continuación el número de la interfaz)	

<table>
<tr><td colspan="2">[Se pone la interfaz que responde al flujo normal de los eventos que se va a describir, en caso que el flujo normal muestre más de una interfaz se pondría a continuación de esta interfaz otra celda con (Interfaz con su número y debajo otra celda con la imagen de la interfaz)
La imagen de la interfaz se enumera con números y con este la referenciamos donde quiera que haga falta dentro de la descripción del CU, le asociamos también un nombre que es el realmente aparecerá en la interfaz cuando se ejecuta el sistema.
Para describir la interfaz se debe poner lo que hace ese control que representa el número asignado, el tipo y el nombre, por ejemplo:
• Control para seleccionar el sexo
Nombre: sexo tipo: select]</td></tr>
<tr><td colspan="2">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>[Aquí se enumeran las acciones que se realizan de forma consecutiva para el flujo normal de eventos.]
[La primera acción para el que la ejecute (ya sea actor o sistema) es como se inicia el CU. Se debe comenzar diciendo “El actor…”. Se realizará la enumeración con números consecutivos, comenzando por el 1. y en la respuesta del sistema se pondría 2. y así paulatinamente]
Para dar paso a las secciones poner antes la viñeta de punto y poner entre paréntesis el nombre de la sección a la que vamos a hacer referencia , ejemplo:
• Si el actor desea modificar (Ver sección 1)</td><td>[En la parte correspondiente a la Respuesta del Sistema debe ponerse en caso de ser necesario el estado en que se muestran los botones de las opciones que se ejecutan, si se muestran o no o si se muestran habilitadas o deshabilitadas, así como los mensajes que muestra.]</td></tr>
<tr><td colspan="2">Cursos alternos</td></tr>
<tr><td colspan="2">[Especificar primeramente en la línea que ocurre este flujo alterno, por ejemplo Línea 4, esto se sobreentiende que este flujo alterno que vamos a poner ocurre si no se realiza lo descrito en el flujo normal]</td></tr>
<tr><td colspan="2">Sección (Nombre de la sección, en caso de que exista)</td></tr>
<tr><td colspan="2">Interfaz (Relacionada con la sección si existe)</td></tr>
<tr><td colspan="2">[Aquí se aplica la explicación dada a inicio de esta plantilla para la interfaz]</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>[Aquí se enumeran las acciones que se realizan de forma</td><td></td></tr>
</table>

consecutiva para la sección se comienza por 1]	
Cursos Alternos (de la sección)	
[Lleva la misma aclaración anteriormente descrita para los flujos alternos del flujo normal]	
Requerimientos No Funcionales	*[De forma general los requisitos no funcionales son asociados a todo el proyecto informático que se está construyendo, para el caso en que haya uno o varios requisitos no funcionales específicos para un caso de uso entonces, especificar su clasificación, poner por ejemplo RNF 1, aclarar que los requisitos no funcionales tienen números en la plantilla de especificación de requisitos]*
Postcondiciones	*[Lo que resulta del caso de uso, condiciones en las que queda el sistema cuando termina la ejecución del CU]*
Prioridad	*[Critico, Importante, Secundario, Auxiliar y Opcional]*

Anexo 6: Plantilla de Cambio de Requisitos

Gestión de cambio en los Requisitos

<Nombre del Proyecto>

<Nombre del producto>

<Versión>

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

1. Introducción

1.1 Propósito

[Resumen del propósito de este documento]

1.2 Alcance

[Breve descripción del alcance de la gestión de cambios en los requisitos]

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

[Lista de documentos a los que se hace referencia]

2. Breve Descripción del Sistema

[Enunciar el objetivo y describir de modo general las principales funcionalidades que tendrá el sistema, incluyendo los módulos (si ya están identificados)]

3. Gestión de Requisitos

3.1 Descripción de los requisitos

[Se realiza una enumeración de todos los requisitos capturados junto a una breve descripción y a su caso de uso asociado]

Doc. Especificación de los Requisitos Versión <x.x>	No.	Requisitos	Descripción	CU_Asociado	Fecha <dd/mm/aa>
	<1>	*[Requisitos capturado en la entrevista con el cliente]*	*[Breve descripción de los requisitos]*	*[Nombre del CU que se asocia al requisitos]*	*[Día, mes y año en que se capturaron los requisitos]*

4. Seguimiento

4.1 Descripción de los cambios en los requisitos

[Lista de requisitos en la matriz de seguimiento, con la descripción correspondiente a cada requisito. Los requisitos son evaluados por los estados siguientes:

- *Nuevo: Requisito que surge mediante un nuevo reajuste de la captura.*
- *Actualizado: Requisito que se ha modificado mediante un nuevo reajuste de la captura.*
- *Eliminado: Requisito que se ha desecho mediante un nuevo reajuste de la captura.*
- *Irrevocable: Requisito que no sufre cambios.*

Nota: Los requisitos que no cambian se colocan en la matriz con su estado correspondiente]

Doc. Matriz de seguimiento Especificación de los Requisitos Versión <x.x>	No	Requisitos	Estado	Descripción	Fecha <dd/mm/aa>	CU_Asociado	Rol Gestor de Cambio
	<1>	*[Requisitos reajustados en la captura en la entrevista con el cliente]*	*[Acción tomada con respecto al requisito]*	*[Breve descripción de los requisitos]*	*[Día, mes y año en que se cambió el requisito]*	*[Nombre del CU que se asocia al requisitos]*	*[Nombre y Apellidos]*

Anexo 7: Plantilla de Especificación de Requisitos.

Especificación de Requerimientos

<Nombre del proyecto>

<Nombre del producto>

<Versión >

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

1. Introducción

1.1 Propósito

[Resumen del propósito de la especificación de requisitos en función para una mejor comprensión]

1.2 Alcance

[Breve descripción del alcance de los requisitos]

1.3 Definiciones, Acrónimos y Abreviaturas

[Breve descripción de las definiciones, acrónimos, y abreviaturas].

1.4 Referencias

[Lista de documentos con los cuales tendrá relación la especificación de requisitos, ejemplo modelo de casos de uso, etc.]

2. Descripción General

[Se describen los factores generales que inciden en el proyecto y sus requisitos. No es el lugar para poner requisitos específicos sino un enfoque general de estos. Pueden incluirse elementos como: Perspectivas del proyecto

- *Funciones generales*
- *Características de los usuarios*
- *Restricciones*
- *Conjeturas y dependencias*
- *Subconjunto de requerimientos generales]*

3. Requisitos Funcionales

[Esta sección describe los requisitos de funcionalidad expresados en lenguaje natural. Puede ser organizada por características comunes, por usuarios o por áreas de trabajo.]

3.1.1 <Primer Requisito de funcionalidad>

[Descripción del requerimiento]

3.1.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

3.1.3 <Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

3.1.4 Estado

[(Propuesto, aprobado)]

3.1.5 Prioridad

[(Critico, importante, secundario)]

3.1.6 Nivel de riesgo asociado a la implementación de la característica

[(Critico, significativo, ordinario)]

3.1.7 Atributos del requerimiento.

4. Requisitos No Funcionales

4.1 Usabilidad

[Esta sección incluye todos los requerimientos que afectan la usabilidad. Por ejemplo:

- *Especificación e entrenamiento necesario para los usuarios en operaciones particulares*
- *Especificación de requerimientos de conformidad con normas y estándares internacionales]*

4.1.1 <Primer requerimiento de usabilidad>

[Descripción del requerimiento]

4.1.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

4.1.3 <Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

4.2 Confiabilidad

[Especificación de requerimientos de confiabilidad. Algunas sugerencias:

- *Especificación del por ciento de tiempo de accesibilidad en horas de uso, mantenimiento, operaciones de desgaste, etc.*
- *Promedio de tiempo entre fallas, generalmente se mide en horas pero puede medirse en días, meses o años.*

- *Tiempo medio de reparación, tiempo promedio que puede demorarse en reparar algún sistema sin provocar un fallo*
- *Rango máximo de defectos*

4.2.1 <Primer requisito de confiabilidad>

[Descripción del requerimiento]

4.2.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

4.2.3 <Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

4.3 Rendimiento

[Se especifican las características de rendimiento por ejemplo:

- *Tiempo de respuesta en una transacción o traspaso de información (Promedio y máximo)*
- *Capacidad, por ejemplo cantidad usuarios permitidos*
- *Recursos tecnológicos requeridos]*

4.3.1 <Primer requerimiento de rendimiento>

[Descripción del requerimiento]

4.3.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

4.3.3 <Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

4.4 Soporte

[Especifica los requerimientos que soportan y mantienen el proyecto. Incluir estándares, convenciones, normas, etc.]

4.4.1 <Primer requisito de soporte>

[Descripción del requerimiento]

4.4.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

4.4.3<Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

4.5 Restricciones de diseño

[Especifica todas las restricciones de diseño para el desarrollo del proyecto]

4.5.1 <Primera restricción de diseño>

[Descripción del requerimiento]

4.5.2 <Precedencia>

[Especificar CU del Negocio con el que se relaciona]

4.5.3 <Seguimiento>

[Especificar CU del Sistema que implementa el requerimiento]

4.6 Requerimiento de ayuda y documentación

[Describe todos los requisitos de ayuda y documentación que requiera el proyecto]

4.7 Adquisición de Componentes

[Incluye todos los componentes que son necesarios adquirir para la ejecución del proyecto]

4.8 Interfaz

[Se especifican los protocolos de comunicación, tipos de interfaces, etc.]

4.8.1 Interfaces de Usuarios

4.8.2 Interfaces con otros Hardware

4.8.3 Interfaces con otros Software

4.8.4 Interfaces de Comunicación

4.9 Requerimientos de licencias y patentes

4.10 Portabilidad

4.11 Políticos culturales

4.12 Legales

4.13 Seguridad

4.14 Hardware

4.15 Aplicación de estándares

Anexo 8: Plantilla de lista de auto-chequeo para el desarrollador.

Auto-chequeo para el desarrollador

Proyecto	Módulo	Etapa	Versión	Fecha
<Nombre >	<Nombre>	<Etapa en que se realiza>	<1.0>	<00/00/00>

Control de Versiones:

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

No.	Evaluación	Eval	NP	Comentario
1	¿Quién utilizará la solución?			
2	¿Hay alguna otra alternativa para la solución que necesita?			
3	¿A qué tipo de problema va dirigida esta solución?			
4	¿Hay alguien más que pueda proporcionar información adicional?			
5	¿Cuantos usuarios van utilizar el sistema? ¿Que puede hacer cada uno de ellos?			
6	¿Los requerimientos funcionales están expresados en un lenguaje comprensible para el cliente/usuario?			
7	¿La estructura y formato de las plantillas facilitan la comprensión?			
8	¿Cada requerimiento admite tan solo una única interpretación?			
9	¿Hay un glosario en el que se define el significado específico de cada término?			
10	¿Los requisitos descritos no presentan ambigüedades?			
11	¿Cada requerimiento se especifica en un único lugar?			
12	¿Cada frase aporta a la especificación?			
13	¿Los requerimientos funcionales son completos y consistentes?			
14	¿Las áreas para las que falta información o es incompleta han sido identificadas?			
15	¿Se ha definido qué información falta?			
16	¿Ningún requerimiento se debería especificar con más (o menos) detalle?			
17	¿Todos los requerimientos están definidos?			
18	¿Algún requerimiento le preocupa o resulta incómodo?			
19	¿Están especificados los cambios posibles a los requerimientos?			
20	¿La probabilidad de cambios está especificada para cada requerimiento?			
21	¿Cada requisito funcional es implementable?			

22	¿Cada requerimiento es relevante para el problema y su solución?			
23	¿El documento de requerimientos está organizado de forma clara y lógica?			
24	¿Cada requerimiento tiene establecido un nivel de prioridad asignado por el Cliente/Usuario para guiar las negociaciones/compromisos?			
25	¿Cada requerimiento es relevante para el problema y su solución?			
26	¿Se identificaron los requisitos funcionales como capacidades o condiciones que el sistema debe cumplir?			
27	¿Los requisitos funcionales especifican la manera en que el sistema debe reaccionar a determinadas entradas?			
28	¿Los requisitos funcionales especifican cómo debe comportarse el sistema en situaciones particulares?			
29	¿Los requisitos funcionales declaran explícitamente lo que el sistema no debe hacer?			
30	¿Los requerimientos funcionales escogen la descripción de todos los servicios esperados por el usuario?			
31	¿Se identificaron los requisitos no funcionales como propiedades o cualidades que el producto debe tener?			
32	¿Los requisitos no funcionales son aplicables al sistema en su totalidad?			
33	¿Se especifican las características que debe cumplir el sistema para ser fácil de utilizar por el cliente/usuario? (requisitos de usabilidad)			
34	¿Se establecen las propiedades que debe tener el sistema para que sea compatible con varios sistemas operativos? (requisitos de portabilidad)			
35	¿Se declaran las características que debe poseer el sistema para que esté disponible a cada uno de los usuarios que lo necesiten? (requisitos de disponibilidad)			
35	¿Se exponen las características que debe tener el sistema para permitir la integridad de la información almacenada? (requisitos de			

	integridad)			
36	¿Se definen las características que debe tener el sistema para lograr su correcto funcionamiento? (requisitos de funcionalidad)			
37	¿Los requisitos funcionales definen propiedades y restricciones del sistema: tiempos de respuesta, requisitos de almacenamiento?			
38	¿Todos los requisitos están declarados en forma de acción (deben terminar en ar, er, ir)?			
39	Se describe quien es responsable del desarrollo de las actividades descritas en el flujo de trabajo de requisito.			
40	¿Hay algo más que debería preguntarme?			

Notas:

<Insertar nota si es necesario>

Leyenda

Eval. : Evaluación que se le otorga al aspecto a evaluar: [0, 2, 4,5,].

NP. : No procede (se marca con una **X** la pregunta que no será necesaria realizarla en el sistema que se le esta aplicando la lista de chequeo).

Comentario: Es obligatorio en las evaluaciones distintas de 5 puntos.

Las evaluaciones serán:

- -	**Malo:** propiedad parcialmente disponible. (0)
-	**Satisfactorio:** propiedad parcialmente disponible. (2)
+	**Bien:** propiedad parcialmente disponible. (4)
++	**Excelente:** propiedad muy bien implementada. (5)

Criterio de evaluación:

( ) Sin modificaciones.
( ) Pequeñas modificaciones.
( ) Grandes modificaciones.
( ) Nueva elaboración.

Conclusión

	Aceptado
	Diferido
	No Aceptado

Anexo 9: Tabla de caracterización de los expertos.

Experto	Graduado de:	Categoría	Labor que realiza	Experiencia
Experto 1	Ing. Ciencias Informáticas.	Instructora	Asesora DDC Ingeniería y Gestión de Software	2 años
Experto 2	Ing. Informático	Aspirante	Jefe de Proyecto	1 años
Experto 3	Ing. Informático	Aspirante	Apoyo al desarrollo de los proyectos	1 años
Experto 4	Ing. Ciencias Informáticas	Adiestrado	Especialista de la Dirección de Calidad de Software	1 años
Experto 5	Ing. Informático	Aspirante a Investigador	Jefe de Proyecto	4 años
Experto 6	Ing. Informático	Adiestrado	Jefe de Proyecto y Arquitecto	2 años
Experto 7	Ing. Informático	-----	Ingeniero Principal	3 años
Experto 8	Ing. Informático	Aspirante Máster	Especialista General de la Dirección de Calidad de la infraestructura Productiva de la UCI	3 años

Anexo 10: Encuesta del coeficiente de conocimiento de los expertos.

Encuesta para determinar el coeficiente de competencia de los expertos

Compañero (a):

En la ejecución de la presente tesis, deseamos someter a la valoración de un grupo de expertos, la propuesta de la mejora de un procedimiento de Gestión de Requisitos aplicándole métricas de calidad, permitiéndonos recoger datos para facilitar la tarea de los líderes de proyectos en la gestión contando con experiencia acumulada de proyectos anteriores. Para ello necesitamos conocer el grado de dominio que UD. posee sobre el proceso; y con ese fin deseamos que responda lo que se le pide a continuación.

Nombre y apellidos: _____

Labor que realiza: _____

Años de experiencia: _____Especialidad:_____

Categoría docente: _____Categoría científica: _____

1.- Marque con una cruz (X) el grado de conocimiento que UD. tiene sobre la temática que se investiga:

0	1	2	3	4	5	6	7	8	9	10

2.- Marque con una cruz (X) las fuentes que le han servido para argumentar el conocimiento que tiene Ud. de la temática que se investiga. Encierre en un círculo la que más ha influido.

No.	**Fuentes de argumentación**	**Grado de influencia**		
		Alto	**Medio**	**Bajo**
1.-	Análisis realizado por Ud.			
2.-	Experiencia.			
3.-	Trabajos de autores nacionales.			
4.-	Trabajos de autores extranjeros.			
5.-	Su propio conocimiento del tema.			
6.-	Su intuición.			

Anexo 11: Encuesta para validar el procedimiento aplicándole métricas de calidad.

Encuesta para validar el procedimiento de mejora para la Gestión de Requisitos aplicándole métricas de calidad

1. ¿Considera usted que es importante para el éxito de los proyectos de software realizar un procedimiento para la Gestión de Requisitos?

Sí___ No___

¿Por qué?_____.

2. ¿Considera usted necesario que el centro UCID adecue el procedimiento definido para desarrollar la Ingeniería de Requisitos?

Sí___ No___

¿Por qué?_____

3. ¿Considera usted que la aplicación del procedimiento propuesto puede ser efectivo para la Gestión de Requisitos del software?

Sí___ No___

¿Por qué?_____.

4. Evalúe el procedimiento propuesto según los siguientes aspectos.

a. Las actividades propuestas son:

___ Innecesarias

___ Necesarias pero no suficientes para desarrollar la Gestión de Requisitos

___ Necesarias y suficientes para desarrollar la Gestión de Requisitos

Otras consideraciones al respecto:_____.

b. La organización de las actividades en el procedimiento propuesto es:

____ Correcta

____ Incorrecta

c. Los artefactos propuestos son:

___ Innecesarios

___ Necesarios pero no suficientes para desarrollar la Gestión de Requisitos

___ Necesarios y suficientes para desarrollar la Gestión de Requisitos

Otras consideraciones al respecto: _____.

d. Para seguir la traza de los requisitos de software la matriz de seguimiento definida en la plantilla de Gestión de Cambios en los requisitos es:

___ Innecesaria

___ Necesaria pero no suficiente

___ Necesaria y suficiente

Otras consideraciones al respecto: _____.

e. ¿Cree eficiente el orden y la unión de las tres técnicas para la captura de requisitos?

Sí___ No___

¿Por qué?_____.

f. La lista de auto-chequeo para el desarrollador es:

____ Correcta

____ Incorrecta

5. Evalué las métricas aplicables al procedimiento según los aspectos siguientes:

5.1 Métricas de la calidad en el proceso de especificación de requisitos**.**

a. La métrica es:

____ Muy adecuada

____ Adecuada

____ Poco adecuada

____ No adecuada

b. Los datos para el cálculo de la métrica son:

___Innecesarios

___ Necesarios pero no suficientes

___ Necesarios y suficientes

Otras consideraciones al respecto: _____.

5.2 Métrica de calidad para la administración de cambio en los requisitos.

a. La métrica es:

____ Muy adecuada

____ Adecuada

____ Poco adecuada

____ No adecuada

b. Los datos para el cálculo de la métrica son:

___Innecesarios

___ Necesarios pero no suficientes

___ Necesarios y suficientes

Otras consideraciones al respecto: _____.

6. Elabore un comentario general sobre el procedimiento que está siendo evaluado que aporte elementos a la mejora del mismo.

_____

_____

_____

_____

_____

_____.

GLOSARIO

Siglas:

BPM: Business Process Management – Procesos de Administración de Negocio.

CMMI: Capability Maturity Model Integration - Integración del Modelo de Madurez de las Capacidades.

CMM: Capability Maturity Model - Modelo de Madurez de las Capacidades.

DFC: Despliegue de la Función de Calidad.

EED: Eficacia en la Eliminación de Defectos.

ERS: Especificación de Requisitos de Software.

GCC: Grupo de Control de Cambio.

GR: Gestión de Requisitos.

IEEE: The Institute of Electrical and Electronics Engineers - Instituto de Ingenieros Eléctricos y Electrónicos.

MINFAR: Ministerio de las Fuerzas Armadas Revolucionarias.

MSF: Microsoft Solution Framework.

RUP: Rational Unified Process - Proceso Unificado del Racional.

SI: Sistema de información.

SP: System Practices - Prácticas Específicas.

SPICE: Software Process Improvement and Capability Determination.

SQA: Software Quality Assurance - Aseguramiento de la Calidad del Software.

SWEBOK: Cuerpo Ingeniería de Software de Conocimiento - Software Engineering Body of Knowledge.

TFEA: Técnicas para facilitar las especificaciones de una aplicación.

TMEC: Tiempo medio entre el cambio.

UCI: Universidad de las Ciencias Informáticas

UCID: Unidad de Compatibilización, Integración y Desarrollo.

UML: Unified Modeling Language –Lenguaje Unificado de Modelado

XP: Extreme Programming.

Términos:

[1] **Proceso:** Conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados, utilizando recursos.

[2] **Software**: Son las instrucciones electrónicas que van a indicar al ordenador que es lo que tiene que hacer. También se puede decir que son los programas usados para dirigir las funciones de un sistema de computación o un hardware.

[3] **Requisito:** Condición o cualidad que debe cumplir alguien o algo.

[4] **Aseguramiento de Calidad:** Conjunto de actividades planificadas y sistemáticas necesarias para proporcionar confianza en que el producto software satisfará los requisitos dados de calidad.

[5] **Necesidades**: No siempre están completamente determinadas y especificadas.

[6] **Entidad:** Todo aquello que se puede describir y considerar individualmente: actividad o proceso, producto, organización o sistema, persona o combinación de todos o algunos de ellos.

[7] **Modelos de Calidad:** En los modelos de calidad, la calidad se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. La calidad es un concepto que se deriva de un conjunto de subconceptos.

[8] **Validación de requisitos:** Actividad para detectar omisiones o ambigüedades en los requisitos.

[9] **Defecto**: Consecuencia de un error. Incumplimiento de un requisito asociado a un uso previsto o especificado.

[10] **Gestión de Calidad:** Determinación y aplicación de las políticas de calidad de la empresa.

[11] **Especificación de Requisitos del Software*:*** Forma de representar los requisitos. Puede ser en un documento o en un gráfico.

[12] **Release**: Versiones del software.

[13] **Stakeholders**: Es cualquier grupo o individuo que pueda afectar o ser afectado por el logro de objetivos de la organización.

[14] **Error:** Acción humana durante el proceso de desarrollo que produce un defecto.

[15] **Juego priorizado:** Conjunto de elementos de carácter primario.