UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD # 6

Título: Procedimiento para definir los requisitos de calidad en el proceso de desarrollo de software en la Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autoras: Liliana Juez Aldana

Alianny Izquierdo Reinoso

Tutora: Msc. Violena Hernández Aguilar

Ciudad de La Habana, Cuba

Junio 2010

La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia.

"John Ruskin"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____días del mes de Junio del año 2010.

Liliana Juez Aldana

_____

Firma del Autor

Alianny Izquierdo Reinoso

_____

Firma del Autor

Msc. Violena Hernández Aguilar

_____

Firma del Tutor

DATOS DE CONTACTO

Tutora: Ing. Violena Hernández Aguilar **Email**: violena@uci.cu

Experiencia profesional:

- Ingeniera Informática, graduada en el 2005 en el Instituto Superior Politécnico José Antonio Echeverría (CUJAE), Ciudad de la Habana, Cuba.
- Graduada de Máster en Gestión de Proyectos Informáticos en la Universidad de las Ciencias Informáticas (UCI) en diciembre del 2009.
- Profesora asistente de la UCI, impartiendo Ingeniería de Software.
- Se desempeña laboralmente como Especialista General de la Dirección de Calidad de la infraestructura Productiva de la UCI.

A nuestro Comandante Fidel Castro Ruz y a la Revolución por darnos la oportunidad de convertirnos en lo que somos hoy.

A Violena, nuestra tutora, por guiarnos y apoyarnos.

A Heydi por ayudarnos a conseguir este tema de investigación.

A todos nuestros familiares y amigos que de una manera u otra hicieron posible este sueño.

A todos los que en esta página no mencionamos pero se encuentran en nuestros corazones.

Lili y Piny

De Liliana

A mis padres por darme la vida y hacer de mí una persona de bien. Gracias por estar siempre a mi lado y por guiarme y levantarme siempre que fallé.

A mi hermana por ser simplemente mi otra mitad y ser siempre un ejemplo a seguir.

A mi esposo Jorge Dayro, por ayudarme a ser quien soy y por estar presente siempre que lo necesité. Te amo.

A Jorge y Orquídea por convertirse en mis segundos padres, y brindarme su apoyo.

A Marilín, Luis y Davicito por brindarme su ayuda y amistad.

A toda mi familia en general, en especial a mi abuelo que ya no se encuentra conmigo.

A todos mis compañeros y amigos que hice a lo largo de toda la carrera que me ayudaron, aconsejaron e hicieron mi vida en la universidad más agradable.

A todos los que no están en esta página pero están en algún lugar de mi corazón.

De Alianny

A mi mamá por ser inspiradora de mi vida, y siempre impulsarme a conseguir lo que quiero ofreciéndome aliento constante para lograrlo.

A mi papá por el apoyo y enseñanzas que siempre me ha dado.

A mi hermana a quien quiero mucho.

A mi novio Ale por ser amigo, compañero, paciente y amante hasta el final. Te amo mucho.

A mi familia por apoyarme durante todo este tiempo.

A todos aquellos amigos que de alguna manera me ayudaron y apoyaron durante estos años y durante el desarrollo de este trabajo, en especial a Ailyn, Ana Niuska, Yaneisy, Ariadna, Themis y otros que aunque no los mencione estarán siempre en mi corazón.

RESUMEN

A pesar del desarrollo alcanzado por la Industria del Software en los últimos años, hoy en día son muchos los proyectos que fracasan, y las causas fundamentales radican en el inadecuado levantamiento de requisitos que se lleva a cabo en los mismos. En el desarrollo de una aplicación informática no todo lo que los clientes van a solicitar es funcionalidad pura. Como parte imprescindible para lograr un software con la calidad que requiere el cliente es indispensable saber seleccionar los requisitos no funcionales y dentro de ellos, los requisitos de calidad.

El presente trabajo está centrado en el desarrollo y aplicación de un procedimiento para definir los requisitos de calidad en el proceso de desarrollo de software en la Universidad de las Ciencias Informáticas. Para esto se realizará un estudio con algunos proyectos para definir como se está realizando la captura de requisitos y así detectar las deficiencias en este proceso.

Como resultado de la investigación se propone un procedimiento para identificar y especificar los requisitos de calidad de los proyectos productivos de la universidad. El procedimiento que propone este estudio favorece a los proyectos ya que contribuye a un correcto levantamiento de requisitos en cualquier proyecto sin importar el tamaño o complejidad.

PALABRAS CLAVE

Calidad, Requisitos no funcionales, Requisitos de Calidad.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, a nivel mundial la Industria del Software se ha convertido en un enorme gigante, el cual crece y se desarrolla a ritmo agitado. Todas las empresas quieren producir aplicaciones informáticas con alta calidad, en el menor tiempo posible y a costos mínimos. Aumentando así la competitividad entre ellas debido a que los clientes son cada vez más exigentes. La calidad del software juega un papel importante dentro del desarrollo de aplicaciones informáticas influyendo positivamente en la decisión de un cliente a la hora de escoger el producto que necesita.

Un estudio realizado por el Grupo Internacional Standish (*Standish Group International*[1]*),* empresa dedicada a estudiar los factores que inciden en el éxito o fracaso de los proyectos de Tecnología de la Información, mostró que el porcentaje de proyectos de software terminados con éxito sigue siendo bajo con relación a estudios realizados en años anteriores, mientras que la necesidad de crear software con calidad sigue aumentando. Destacando además que uno de los principales factores que inciden en el fracaso de los mismos son problemas relacionados a los requisitos.

Una de las tareas de mayor importancia en el ciclo de vida del desarrollo de software es la obtención de requisitos, ya que con ello se determinará la base de la nueva aplicación. Todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos.

Los requisitos definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Los requisitos funcionales (RF) son las condiciones que el sistema debe cumplir, mientras que los requisitos no funcionales (RNF) son cualidades que el producto debe tener.

Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con

[1] **Standish Group International:** Empresa dedicada a obtener información de los proyectos fallidos de Tecnología de la Información, con el objetivo de encontrar y combatir las causas de los fracasos.

poca aceptación. Una buena definición de requisitos no funcionales es tan importante como los requisitos funcionales. Estos deben ser apropiadamente definidos, analizados y trazados.

Dentro de los requisitos no funcionales están los requisitos de calidad, los cuales juegan un papel importante dentro de este grupo. Uno de los pasos para garantizar que el producto sea óptimo es identificar y entender qué atributos de calidad se deben tener en cuenta a la hora de desarrollar los requisitos no funcionales.

La Universidad de las Ciencias Informáticas, creada sobre una fuerte base tecnológica y un amplio perfil productivo aspira convertir la industria cubana del software en un renglón fundamental de la economía, ser líder del desarrollo de las empresas de software en Cuba e insertarse en el mercado internacional. En sus laboratorios de calidad se realizan constantemente pruebas y revisiones a los proyectos que se desarrollan en ella para controlar la calidad con la que se van desarrollando. Pero a pesar de todos los esfuerzos realizados, muchos no salen en tiempo y no cumplen adecuadamente con las expectativas de los clientes.

Muchos de esos proyectos fallidos han presentado problemas para obtener una buena especificación de los requisitos de calidad, puesto que no se tiene una visión clara de ello. En muchos casos se pasa por alto la selección de estos, lo cual ha traído consigo que algunas funcionalidades del producto final se afecten. Se tiene muy poco conocimiento en el tema de los requisitos de calidad ignorando la importancia que tienen en el desarrollo de un software. En la universidad no existe bibliografía que muestre clasificación alguna de este tipo de requisitos.

Por todo lo anteriormente planteado surge el **problema** que se pretende resolver: ¿Cómo identificar y especificar los requisitos de calidad que garanticen la disminución de los riesgos asociados con una débil solución arquitectónica?

El **objetivo general** es desarrollar un procedimiento que establezca como identificar y especificar los requisitos de calidad de un proyecto.

El **objeto de estudio** está enmarcado en la Ingeniería de Requisitos.

El **campo de acción** está definido por los requisitos de calidad de software en los proyectos productivos de la universidad.

Como **objetivos específicos** se plantean:

- Diseñar un procedimiento que establezca como identificar y especificar los requisitos de calidad de un proyecto.
- Aplicar el procedimiento en un proyecto real de la universidad.
- Validar el procedimiento mediante el panel de expertos.

Para el logro de los objetivos se trazaron las siguientes **tareas**:

- Estudio de la bibliografía existente y estado del arte relacionado con la Ingeniería de Requisitos.
- Análisis del proceso de selección de requisitos de calidad en los proyectos de la universidad mediante el uso de métodos empíricos.
- Descripción del proceso de selección de requisitos de calidad.
- Desarrollo de un procedimiento para identificar y especificar los requisitos de calidad de un proyecto.
- Proposición de una lista de chequeo para identificar y priorizar requisitos de calidad.
- Proposición de una plantilla donde se describa como se debe especificar los requisitos de calidad.
- Aplicación del procedimiento en un proyecto de la universidad.
- Validación mediante el panel de expertos.

Para guiar la investigación se plantea como **idea a defender** que la definición de un procedimiento que establezca como identificar y especificar los requisitos de calidad de un proyecto contribuirá a asegurar la calidad de los productos de software en los proyectos productivos de la universidad.

Para llevar a cabo la investigación se emplearon diferentes métodos científicos, los cuales posibilitaron el avance en el conocimiento del objeto de estudio logrando el objetivo planteado.

Métodos científicos empleados

Métodos empíricos:

Entrevista: Dirigida a profesores y especialistas en esta área integrados a los proyectos de la universidad con el propósito de conocer los puntos que siguen para definir los requisitos no funcionales de un proyecto y dentro de estos los requisitos de calidad.

Encuesta: Se realizan encuestas a estudiantes y profesores de proyectos productivos para comprobar el conocimiento que tienen sobre el tema, y así comprobar los problemas existentes en la captura de los requisitos no funcionales.

Métodos teóricos:

Histórico-lógico: Permitió hacer un estudio de la trayectoria histórica real de la forma en que se definen los requisitos no funcionales de un proyecto y dentro de estos los requisitos de calidad, así como su evolución y desarrollo. Poniendo atención en las dificultades que se presentan y los errores que se cometen.

Analítico-Sintético: Análisis de la documentación disponible de proyectos anteriores para hacer un estudio lo más abarcador posible del estado del problema a resolver y extraer los elementos más importantes que influyen en el logro del objetivo.

El presente trabajo se encuentra estructurado por un resumen, introducción, tres capítulos en los cuales se expone todo el trabajo investigativo y práctico realizado, conclusiones generales, bibliografía y anexos.

Capítulo 1. Fundamentación teórica: En este capítulo se realiza un análisis del estado del arte de los procesos que son objetos de estudio y se argumenta la relación existente entre ellos. Se abordan los principales conceptos asociados a los requisitos de calidad que serán de gran importancia para la comprensión de lo que se describe en el resto del trabajo. Se hace un estudio en la universidad sobre la captura de requisitos no funcionales, y dentro de ellos los requisitos de calidad.

Capítulo 2. Solución propuesta: Ya en el capítulo dos se analiza y describe la solución propuesta. Se establecen además un conjunto de artefactos que guían el proceso de identificación y especificación de los requisitos de calidad dentro de cualquier proyecto de la universidad.

Capítulo 3. Validación de la solución propuesta: Por último se valora la utilidad de la solución propuesta a partir de la situación actual aplicando el procedimiento en varios productos ya realizados por la universidad y validándose mediante el panel de expertos.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción del capítulo

En este capítulo se fundamenta el estado del arte a través de la revisión bibliográfica realizada, donde se brinda una visión general de la captura de los requisitos. Se abordan temas relacionados con la Calidad de Software y la Ingeniería de Requisitos, destacando en todo momento la importancia de estos dentro de un proyecto. Se profundiza en los requisitos no funcionales y los requisitos de calidad. Además se abarcan los conceptos fundamentales que sirven de soporte teórico a la investigación desarrollada y que estarán presentes a lo largo de este trabajo de diploma.

1.2 Situación de la Industria del software

1.2.1 A nivel mundial

Estudios realizados por varias empresas a nivel mundial han demostrado la existencia de problemas en el desarrollo de software de forma general. Desde 1994, el *Standish Group International* realiza estudios en los que se encuesta a directores de proyectos sobre la situación y problemas de los mismos. En el 2009 sus estudios mostraron que los proyectos de software tienen una tasa de éxito del 32%. Por otra parte el 44% de los proyectos fueron impugnados, mientras que el 24% fracasó. En la figura 1 se muestran los por cientos de los proyectos fracasados, cuestionados y con éxito en el año 2009 según el Standish. [1]

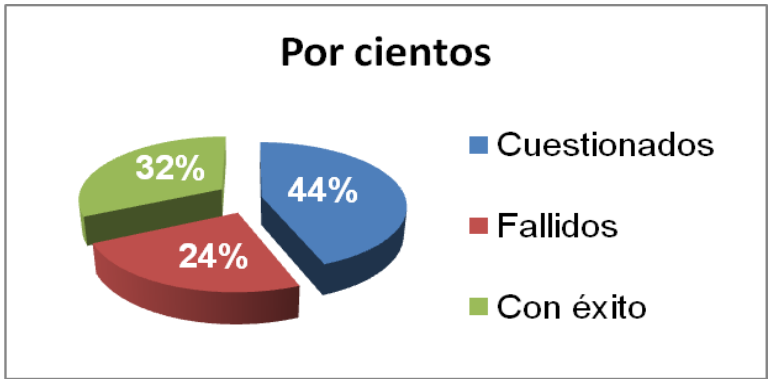

Figura 1. Por ciento de proyectos con éxito, fallidos y cuestionados

Fallidos: cancelados antes de su finalización o entregados y nunca utilizados.

Con éxito: entregados a tiempo, sobre el presupuesto, con las características requeridas.

Cuestionados: tarde, presupuestado, y / o con menos de las características requeridas de las funciones.

La siguiente tabla (tabla 1) muestra una comparación entre los años del 1994 al 2009 respecto a los proyectos con éxito, cuestionados y fallidos. [1]

Año / Calificación	2009	2006	2004	2002	2000	1998	1996	1994
Con éxito	32%	35%	29%	34%	28%	26%	27%	16%
Cuestionados	44%	19%	53%	15%	23%	28%	40%	31%
Fallidos	24%	46%	18%	51%	49%	46%	33%	53%

Tabla 1. Comparación entre años del resultado de proyectos con éxito, cuestionados o fallidos.

Si se analizan los resultados obtenidos años atrás con el año 2009 que se muestran en la tabla 1, se puede observar que en este año ha habido una mejoría, pero las cifras de los proyectos con éxito siguen siendo bajas, mientras que la competencia entre compañías sigue aumentando, así como la necesidad de un producto con calidad. En la figura 2 se muestran las principales causas que llevan a los proyectos al fracaso según dicho estudio. [1]

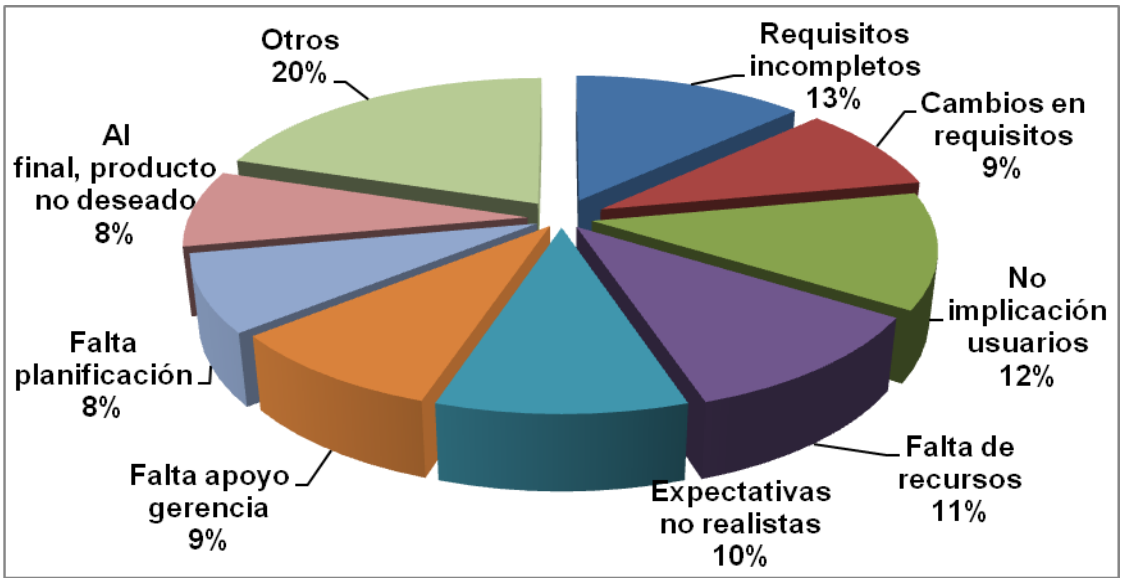

Figura 2. Causas por las cuales fracasan los proyectos [Standish Group International]

Como se puede observar en la figura 2, factores como cambios en los requisitos, requisitos incompletos y expectativas no realistas están estrechamente vinculados al levantamiento de requisitos.

En los últimos años de la industria del software se han ido definiendo modelos basados en experiencias de la Ingeniería de Software sirviendo de guía para las mejoras y unificando los criterios de evaluación para las empresas. Las normas ISO, el modelo estadounidense Modelo de Madurez y Capacidad (CMM - Capability Maturity Model), el Estándar Europeo para Evaluación y Mejoras de Procesos de Desarrollo de Software (BOOTSTRAP) son algunos de los modelos que se destacan. Y aún con la existencia de metodologías, estándares, normas, técnicas y herramientas para lograr un mejor resultado en las aplicaciones informáticas, muchos sistemas siguen fallando, no por falta de presupuesto o tecnología, sino porque se siguen cometiendo errores en el levantamiento de requisitos.

1.2.2 En Cuba

La sociedad cubana crece en un ambiente donde la tecnología es el centro de un mundo dinámico y aunque no posee los recursos necesarios pretende informatizar el país e insertarse en la industria del software con el fin de desarrollar su economía. Desde 1997 se considera la informática como un renglón económico generador de ingresos para el país.

“La Industria Cubana del Software (ICSW) está llamada a convertirse en una significativa fuente de ingresos nacional, como resultado del correcto aprovechamiento de las ventajas del considerable capital humano disponible.” [MINREX 2004] [2]

Por tal motivo se han creado empresas que cumplen con diferentes funciones relacionadas con la producción, mantenimiento, venta y prestación de servicios de software. Algunas de ellas son: Empresa Productora de Software para la Técnica Electrónica (SOFTEL), Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados CITMATEL. En el 2002 surge la Universidad de las Ciencias Informáticas que tiene dentro de sus prioridades la producción de soluciones tecnológicas integrales y servicios de software para la exportación y la paulatina informatización del país.

Se han creado también normas para medir la calidad de estos servicios. Ejemplo de ello es la norma (NC) ISO\IEC 9126 adaptada a los proyectos del país. Esta norma cubana permite especificar y evaluar la calidad del producto de software desde las perspectivas de aquellos asociados con la adquisición, regulación, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoría del software. [3]

1.3 Calidad de software: un indicador para asegurar y evaluar el éxito del proyecto

En el mundo globalizado de hoy, donde las organizaciones se ven enfrentadas a competencia de nivel mundial, la calidad surge como una necesidad y se convierte en un importante punto diferenciador, aumentando la satisfacción general del cliente, disminuyendo costos y optimizando recursos. Los compradores prefieren productos o servicios que ostentan certificados de calidad porque les transmite seguridad y confianza.

La calidad de software es todo el conjunto de cualidades que lo caracterizan determinando su eficiencia y utilidad, satisfaciendo las necesidades tanto implícitas como explícitas del cliente. La IEEE.Std.610-1990 la define como el grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades o expectativas del cliente o usuario. [IEEE.Std.610-1990] [4]

Según Roger S. Pressman, ingeniero de software, profesor, consultor y autor de productos centrados en la Ingeniería del Software, la calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente. [Pressman, 1998] [5]

En la presente investigación se concuerda con la definición anterior ya que el autor destaca un punto importante en la calidad de software: los requisitos, siendo estos la base fundamental para lograr la calidad del producto final. La calidad de software va a depender en su totalidad de la concordancia entre los requisitos planteados respecto a los obtenidos. Ambos conceptos resaltan la necesidad de que un software de calidad debe satisfacer los requisitos dados por el usuario.

1.3.1 Modelos, estándares y métricas de calidad

Hoy en día ha aumentado de manera considerable la complejidad de los problemas que buscan una solución en el software. Las organizaciones quieren ser capaces de producir software confiable, a tiempo y dentro del presupuesto acordado con el cliente, mientras que estos quieren saber si todo lo anterior se cumplirá. Como consecuencia, surgen los modelos y estándares de calidad para ayudar a las compañías productoras de software a conseguir la calidad de sus productos.

Un modelo de calidad de software es un conjunto de buenas prácticas para el ciclo de vida del software, enfocadas en los procesos de gestión y desarrollo de proyectos. Este indica qué hacer y no cómo hacerlo.

Según la NC-ISO/IEC 9126-1 es el conjunto de características y las relaciones entre las mismas, que proveen la base para especificar requisitos de calidad y evaluar la calidad. [NC-ISO/IEC 9126-1, 2005] [3]

El doctor Witold Suryn, profesor en la escuela de Tecnología Superior, en Montreal, Canadá; expresa que los modelos de calidad presentan un acercamiento para unir diferentes atributos de calidad con los objetivos básicos de: ayudar a entender como varias facetas de calidad contribuyen al todo y enfatizar claramente que la calidad de software es mucho más que simplemente defectos y fracasos. Además los modelos representan una ayuda para navegar por el mapa de características de calidad, subcaracterísticas y medidas apropiadas y por último, ayudan a definir el perfil de evaluación. [6]

Por su parte, los **estándares de calidad** permiten la definición de un grupo de criterios de desarrollo que guiarán el proceso en la producción de un software. Estos proporcionan un marco para implementar procedimientos de aseguramiento de la calidad.

1.3.2 Análisis de modelos y estándares de calidad

Pressman (2002) indica que los factores que afectan a la calidad del software no cambian, por lo que resulta útil el estudio de los modelos de calidad que han sido propuestos en este sentido desde los años 70. En este trabajo se estudiarán algunos de los modelos más importantes propuestos hasta el momento: Dromey (1996), la ISO/IEC 9126 y CMMI.

- **Modelo Dromey (Robert Dromey, 1996)**

Uno de los modelos de calidad más antiguos es el modelo McCall, presentado en 1977. Este modelo se centra en el producto final, identificando atributos claves denominados atributos de calidad desde el punto de vista del usuario. De este modelo se derivaron otros como el modelo Dromey.

Dromey (1996) propuso un marco de referencia para la construcción de modelos de calidad, basado en cómo las propiedades medibles de un producto de software pueden afectar los atributos de calidad generales, como por ejemplo, confiabilidad y mantenibilidad. El problema que se plantea es cómo conectar tales propiedades del producto con los atributos de calidad de alto nivel. Para solventar esta situación, Dromey sugiere el uso de cuatro categorías que implican propiedades de calidad, que son: correctitud, internas, contextuales y descriptivas. [7] La siguiente figura (figura 3) muestra la relación que establece Dromey entre las propiedades de calidad del producto y los atributos de calidad de alto nivel.

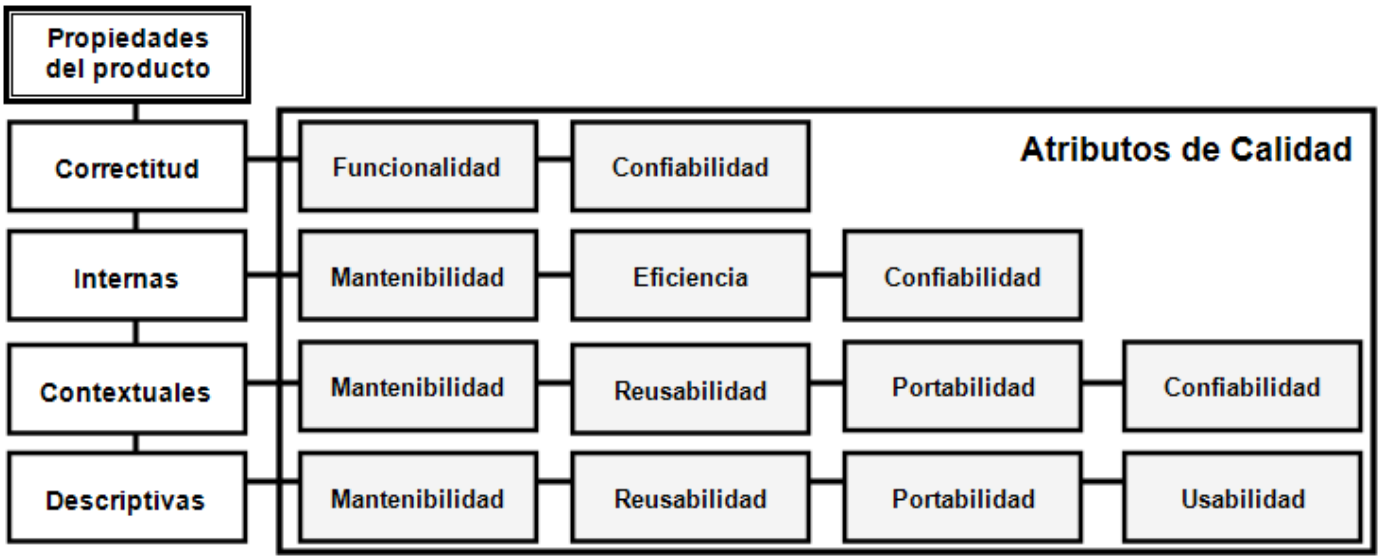

Figura 3. Criterios asociados a los factores de Calidad. [Modelo Dromey (1996)]

Dromey propone un proceso de construcción de modelos de calidad que consta de cinco pasos basados en las propiedades mencionadas. Él demuestra el uso de su procedimiento para la construcción de un modelo de calidad de implementación, un modelo de calidad de requisitos, y un modelo de calidad de diseño. [7] La importancia de este modelo es que, a diferencia de los anteriores, plantea que no todos los productos de software pueden evaluar su calidad de la misma manera, por lo que no deberían verse obligados por la necesidad de basarse en un mismo modelo general como postula la ISO.

Con este modelo se pueden observar los atributos de calidad propuestos por el autor que reflejan la calidad del software. Los **atributos de calidad** son entidades que pueden ser verificadas o medidas en el software según la IEEE. [ISO/IEC 9126]

El profesor Ian Sommerville de la universidad St. Andrews en Escocia, conocido por su libro de referencia de la Ingeniería de Software, opina que los atributos reflejan el comportamiento del software durante su ejecución y en la estructura y organización del programa fuente y en la documentación asociada. [Sommerville, 2005] [8] Por lo que se puede concluir que los atributos de calidad son características que sirven para medir un software.

- **ISO 9126 Software de evaluación de productos: Características de calidad y directrices para su uso (Software Product Evaluation: Quality Characteristics and Guidelines for their Use)**

La ISO/IEC 9126 es un estándar internacional para la evaluación del software. Es supervisado por el proyecto SquaRE (Ingeniería de Requisitos de Calidad de Seguridad o Security Quality Requirements Engineering) y la ISO 25000:2005, que siguen los mismos conceptos generales. Este surge debido a la necesidad de un modelo único para expresar la calidad de un software.

El estándar se divide en cuatro partes que trata, respectivamente, los temas siguientes: *modelo de la calidad, métricas externas*, *métricas internas*, y *métricas de calidad en uso* (figura 4). El modelo de calidad que propone este estándar se divide en dos partes: calidad externa y calidad interna, y la calidad durante el uso. La primera parte del modelo especifica seis características para la calidad interna y externa, que son además divididas en sub-características y son el resultado de los atributos o cualidades internas del software. La segunda parte del modelo especifica cuatro características de calidad durante el uso del producto [NC-ISO/IEC 9126-1, 2005].

La calidad de cualquiera de los procesos del ciclo de vida, contribuye a mejorar la calidad del producto, y esta a su vez contribuye a mejorar la calidad en el uso. Por consiguiente, evaluar y mejorar un proceso es un medio para mejorar la calidad del producto; la evaluación y mejora de la calidad del producto son una vía para mejorar la calidad durante el uso (figura 4). De igual modo, la evaluación de la calidad durante el uso permite la retroalimentación para mejorar un producto, y cuando se produce la evaluación permite la retroalimentación para mejorar un proceso [NC-ISO/IEC 9126-1, 2005]

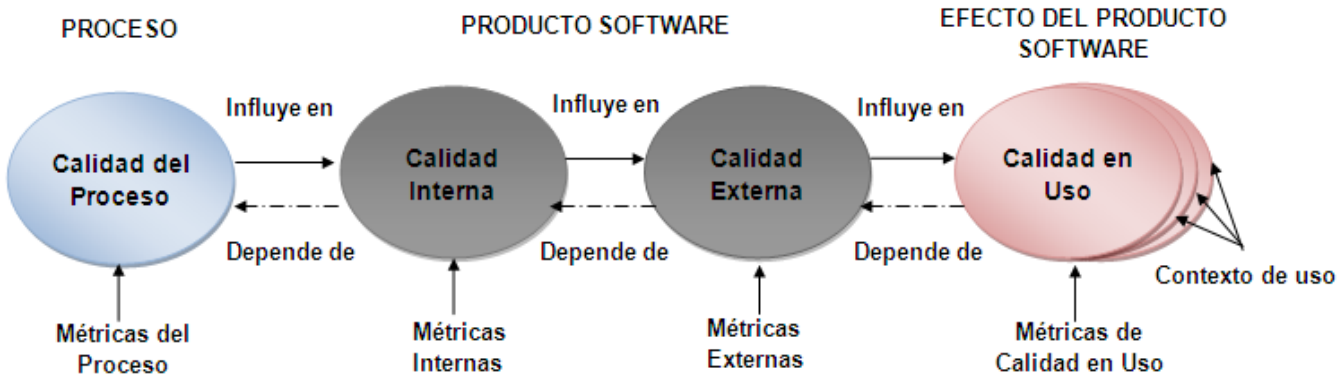

Figura 4. Relación entre los diferentes enfoques hacia la calidad [ISO/IEC 9126]

Como muestra la figura 4, las métricas internas pueden ser aplicadas a los productos intermedios que se desarrollan a lo largo del ciclo de vida de desarrollo de un software. Estas le proporcionan a los desarrolladores la habilidad de medir la calidad de estos productos intermedios, con lo cual se puede predecir la calidad del producto final. [ISO/IEC 9126-2, 2003]

Por su lado, las métricas externas pueden ser usadas para medir la calidad del producto de software a través de la medición del comportamiento del sistema del cual el software forma parte (figura 4). Por último, las métricas de calidad en uso (figura 4) miden si un producto resuelve las necesidades de usuarios específicos para alcanzar metas específicas con eficacia, productividad, seguridad y satisfacción en un contexto específico de uso. Esto solo puede lograrse en un entorno real del sistema.

Este modelo se ha desarrollado en un intento de identificar los atributos más importantes para la calidad interna y externa en un producto software. Este identifica seis características claves de calidad donde cada una de ellas se descompone en un conjunto de sub-características como se muestra en la figura 5. [NC-ISO/IEC 9126-1, 2005]

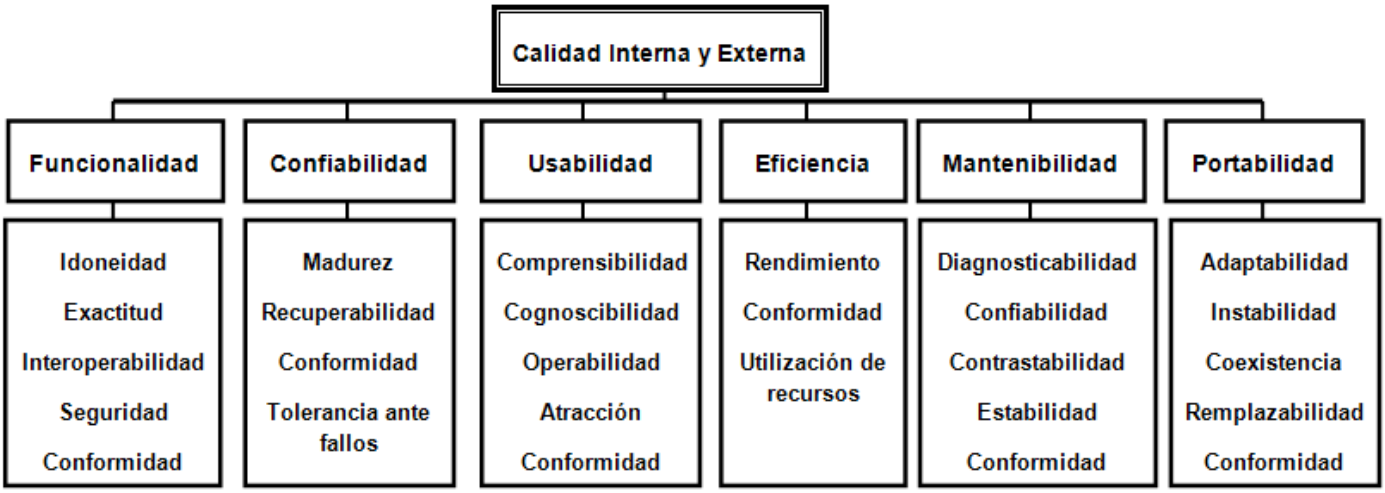

Figura 5. Modelo ISO/IEC 9126.

Los factores de calidad que contempla el estándar ISO/IEC 9126 no son necesariamente usados para mediciones directas pero proveen una valiosa base para medidas indirectas, y una excelente lista para determinar la calidad de un sistema. [9]

En este caso, la ISO 9126 llama características de calidad a lo que el modelo Dromey nombraba como atributos de calidad. Diversas bibliografías hacen referencia a lo mismo pero con distintos nombres. Para

una mejor comprensión en el presente trabajo se nombrarán como características y subcaracterísticas de calidad como lo hace el estándar 9126.

A diferencia del modelo Dromey, la ISO/IEC 9126 es más completa ya que propone un modelo de calidad dividido en calidad interna y externa y calidad en uso, identificando además los atributos de calidad que considera más importantes. A pesar de los pasos de avance que se ve con este modelo, no se observa aún que se especifiquen los requisitos de calidad.

Luego de estudiar las características de calidad propuestas por la ISOI\IEC 9126 y las propuestas por el modelo Dromey, se tomarán como características y subcaracterísticas de calidad las que se muestran en la siguiente figura (figura 6). En torno a estas girará el procedimiento que se propone en el siguiente capítulo. En el anexo 1 aparecen explicadas cada una de las características y subcaracterísticas de calidad.

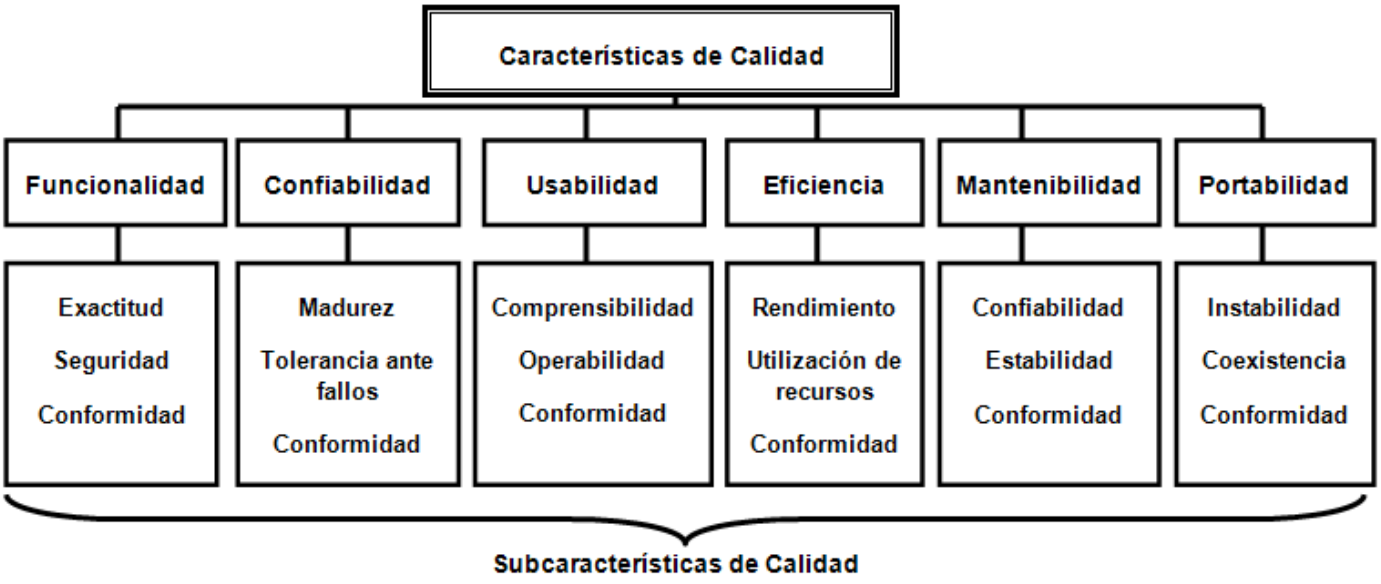

Figura 6. Características y subcaracterísticas de Calidad escogidas para el procedimiento.

- **CMMI: Modelo de Madurez y Capacidad Integrado**

El Modelo de Madurez y Capacidad Integrado (CMMI: Capability Maturity Model Integration) se presenta como un modelo de mejora de procesos en el desarrollo de software que puede complementarse a la norma ISO 9001. Fue desarrollado por el Instituto de Ingeniería de Software (SEI: Software Engineering Institute) de la Universidad Carnegie Mellón y publicada su primera versión en el año 2000. Tiene como

objetivo proveer una guía para mejorar los procesos de una organización y la capacidad para gestionar el desarrollo, la adquisición y el mantenimiento de productos y servicios. Además describe un conjunto de buenas prácticas, tanto de gestión como de ingeniería. [10]

Los elementos principales del CMMI son las áreas de proceso, o sea, procesos que se realizan en una organización para desarrollar productos y servicios. Aunque los requisitos afectan muchas áreas de proceso, CMMI define dos áreas expresamente relacionadas con los requisitos: Gestión de requisitos (GR: Requirements Management) en el nivel dos de la representación escalonada y Desarrollo de requisitos (DR: Requeriments Development) en el nivel tres como se muestra en la figura 7.

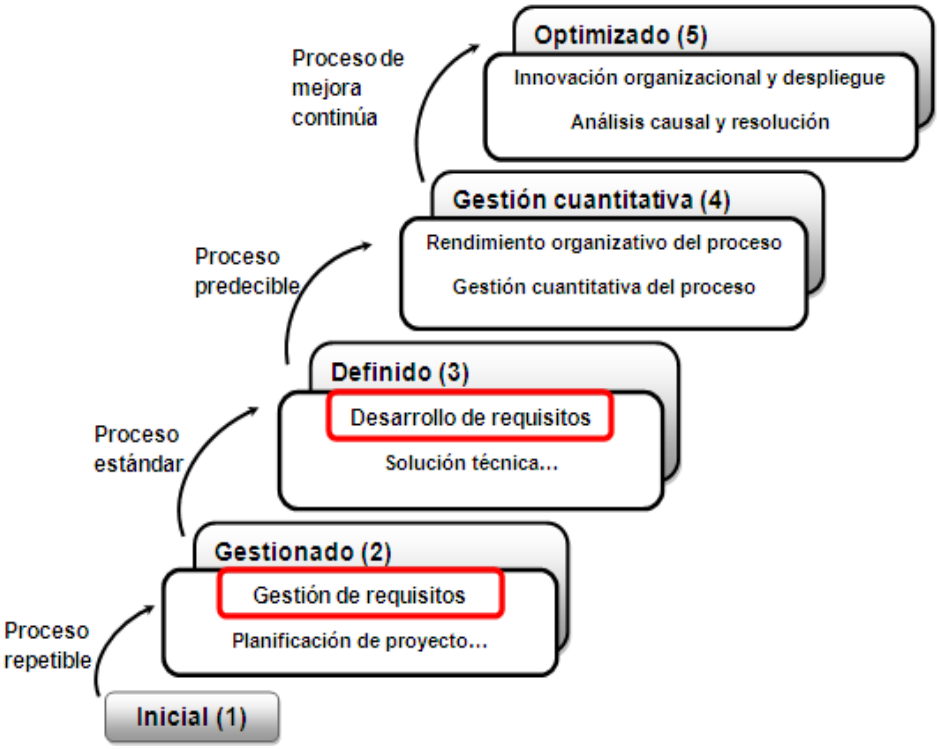

Figura 7. Representación escalonada de las áreas de proceso de CMMI.

El objetivo fundamental de la Gestión de Requisitos es la gestión de los requisitos del proyecto y la identificación de inconsistencias de los requisitos con los planes del proyecto y con los productos de trabajo del proyecto. Las metas específicas de esta área de proceso son las de obtener y analizar los requisitos, obtener la aprobación de éstos, gestionar los cambios de los requisitos, mantener la trazabilidad bidireccional de los requisitos entre sus fuentes y los artefactos que se derivan de ellos e identificar inconsistencias entre el desarrollo y los requisitos.

El área de proceso de Desarrollo de Requisitos tiene como principal objetivo analizar los requisitos del cliente y extraer a partir de ellos los requisitos de la aplicación que se ha de desarrollar. En esta área de proceso se incluye el análisis de los requisitos de los clientes, construyendo un modelo del dominio del problema y relacionando los requisitos de los clientes con dicho modelo, para seleccionar y desarrollar los requisitos que han de ser soportados por el producto o aplicación. Después se construye una especificación de requisitos que permita la comprensión de las funciones del mismo, que incluya los requisitos de interfaz, y que permita la validación de requisitos. [11]

Aunque CMMI tiene dos áreas de procesos para gestionar los requisitos no incluye ninguna definición o especificación para los requisitos de calidad. Al igual que otros modelos trata a los requisitos de calidad como requisitos funcionales y no funcionales. El Dr. Suryn opina que los modelos de calidad son buenos pero estos solo presentan la opinión de sus autores, no llegan a un acuerdo general de comunidad, no tienen medidas asociadas a ellos y en ocasiones son inútiles en la práctica y desconocidos en la industria del software. [6]

Luego de analizar estas normas, se concuerda con la opinión del Dr. Suryn. Se concluye además que ninguna de las normas estudiadas establece el modo en que se ha de determinar los requisitos de calidad relevantes para el producto a construirse. Elicitar los requisitos de calidad aparenta ser una tarea fácil pero podría resultar ser engorrosa y propensa a errores si no se tiene establecido un procedimiento que indique la forma correcta de realizar esta actividad.

1.4 La calidad y los requisitos de software

Aún con la existencia de normas y estándares de calidad se siguen produciendo software que no cumplen con las expectativas de usuarios y clientes, indicando que los principales problemas residen en las primeras etapas de desarrollo. Llegar a entender los requisitos de un problema se considera una de las tareas más difíciles a las que se enfrenta un ingeniero de software actualmente. Si no se realiza una captura de requisitos con calidad, el software resultante tendrá una alta probabilidad de no satisfacer las necesidades del cliente. Todos estos problemas convencen cada vez más la necesidad de mejorar la Ingeniería de Requisitos.

Pressman define la Ingeniería de Requisitos como la ayuda a los ingenieros de software para entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender

cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software. [Pressman, 2006] [12]

Sommerville define la Ingeniería de Requisitos como el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema". [Sommerville, 2005]

Se concluye que la Ingeniería de Requisitos no es más que una rama de la Ingeniería de Software que provee al ingeniero una vía para comprender las tareas relacionadas con la determinación de las necesidades o condiciones del sistema. Su objetivo principal es detallar el software para hacer una especificación completa de este.

La Ingeniería de Requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad y negociar una solución razonable con el mismo. Esta rama de la ingeniería cumple un papel primordial en el proceso de producción de software, ya que se enfoca en un área fundamental: la definición de lo que se desea producir. Por todo lo anteriormente planteado, se considera que la Ingeniería de Requisitos representa una etapa fundamental durante el desarrollo de un producto, pues ayuda a entender mejor el problema proporcionando una guía para alcanzar los objetivos trazados.

El proceso de Ingeniería de Requisitos es un conjunto estructurado de actividades de cuya ejecución se obtiene, valida y mantiene el documento de requisitos del sistema. El proceso define las actividades a realizar, su secuencia, entradas y salidas de cada actividad. Las actividades a realizar son: identificación, análisis, especificación, verificación y validación y gestión. De estas actividades se hará énfasis solo en la identificación y especificación de los requisitos. [13]

Especificación de requisitos

La especificación se centra en la organización de los requisitos capturados. Es la descripción detallada del comportamiento de la aplicación a desarrollar. Una Especificación de Requisitos es una parte importante del proceso de requisitos del ciclo de vida de software y se usa en el diseño, aplicación, supervisión, comprobación, aprobación y pruebas como está descrito en la IEEE Std. 1074-1997. [14]

Según el Proceso Unificado de Desarrollo de Software (RUP: Rational Unified Process) la Especificación de Requisitos de Software (ERS) es el producto resultante sobre los requisitos e inicialmente se considera

en la fase de Inicio como un complemento para definir el alcance del sistema, luego se refina en un modo incremental en las fases de Elaboración y Construcción. [RUP, 2003] [15]

Los principales objetivos que se identifican en la ERS son: ayudar a los clientes a describir claramente lo que se desea obtener mediante un determinado software, ayudar a los desarrolladores a entender qué quiere exactamente el cliente y servir de base para el desarrollo de estándares de ERS particulares para cada organización. [Chalmeta, 2000] [IEEE 830-1998] [16]

Para afirmar que una Especificación de Requisitos se ha realizado con calidad, debe cumplir con una serie de características o condiciones. Según la IEEE 830-1998, una especificación debe ser correcta, inequívoca, completa, consistente, comprobable, modificable, identificable y se debe delinear que tiene importancia y/o estabilidad. Un factor importante para lograr este objetivo es el entendimiento entre ambas partes mediante la comunicación.

Una buena Especificación de Requisitos de Software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores, la reducción del esfuerzo en el desarrollo, una buena base para la estimación de costos y planificación, un punto de referencia para procesos de verificación y validación y una base para la identificación de posibles mejoras en los procesos analizados.

Documento de Especificación de Requisitos

El documento de Especificación de Requisitos es la declaración oficial de qué deben implementar los desarrolladores del sistema [Sommerville, 2005]. En este documento se plasma todo lo que es requerido para que el sistema sea desarrollado, incluyendo la definición y especificación de los requisitos. No es un documento de diseño, es un conjunto de lo que es el sistema y como lo hará. Este documento debe ser organizado para que los cambios que se le hagan a los requisitos puedan ser hechos sin tener que re-escribir demasiado. Además, se puede considerar una herramienta de gran importancia para clientes y desarrolladores, ya que muestra toda la descripción de lo que el sistema hará.

1.4.1 Requisitos del software

Los requisitos se han convertido en un punto clave en el desarrollo de las aplicaciones informáticas. Un gran número de proyectos de software naufragan debido a una mala definición, especificación o administración de requisitos. Factores tales como requisitos incompletos o mal manejo de los cambios de los requisitos llevan a proyectos completos al fracaso total. El Glosario de Terminología Estándar de

Ingeniería de Software (IEEE: Standard Glossary of Software Engineering Terminology) define al requisito como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

3. Una representación en forma de documento de una condición o capacidad como las expresadas en 1 o en 2. [IEEE.Std.610-1990]

El profesor Sommerville presenta una definición acerca de lo que es un requisito: “Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.” [Sommerville, 2005]

Por las definiciones anteriormente consideradas se concluye que un requisito no es más que las diferentes características y funcionalidades que un sistema debe tener para darle solución a un problema dado.

Debido a que los requisitos son las necesidades del producto que se debe desarrollar en cualquier proyecto de software, es importante no perder de vista que un requisito debe ser especificado por escrito como todo contrato o acuerdo entre dos partes; posible de probar o verificar para poder comprobar si se cumplió con él o no; consistente que no entre en contradicción con otros requisitos y conciso, o sea, fácil de leer y entender. Además, un requisito deber estar completo, es decir, que proporcione la información suficiente para su comprensión. Y por último no debe ser ambiguo para no causarle confusiones al lector. [IEEE 830-1998]

Se deben tener en cuenta estas características si se quiere obtener un producto con calidad que cumpla con las expectativas del cliente. Los requisitos del software son los fundamentos desde los que se mide la calidad. Requisitos ambiguos, mal especificados o con los problemas antes mencionados se consideran una falta grave de calidad.

Clasificación de los requisitos

El profesor Sommerville divide los requisitos en dos grupos: requisitos del usuario para designar los requisitos abstractos de alto nivel y los requisitos del sistema para designar la descripción detallada de lo que el sistema debe hacer, tal como se muestra en la figura 8. [Sommerville, 2005]

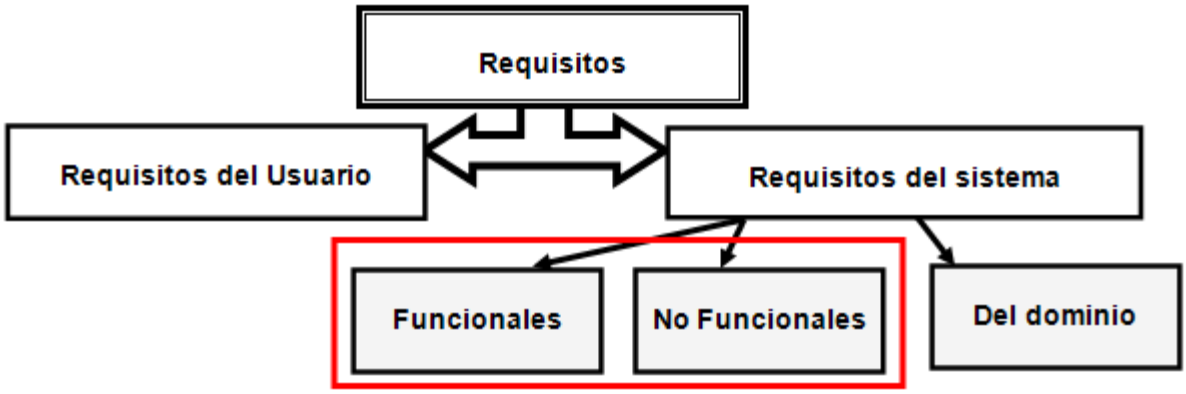

Figura 8. Clasificación de los requisitos según Sommerville [2005].

Luego desglosa los requisitos del sistema de software en funcionales y no funcionales o del dominio (figura 8). Los requisitos funcionales son los servicios que el sistema debe proporcionar, mientras que los requisitos no funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema. Los requisitos del dominio provienen del dominio de aplicación del sistema y reflejan las características y restricciones de ese dominio (pueden ser funcionales o no funcionales). [Sommerville, 2005]. Se tendrá en cuenta esta clasificación dada por Sommerville ya que en ella separa a los requisitos funcionales y los no funcionales. Más adelante se hará énfasis en los requisitos no funcionales.

Defectos más comunes en los requisitos y sus consecuencias

A diario, en los proyectos desarrolladores de software, por falta de información o de conocimiento acerca del tema, se cometen errores con respecto a los requisitos. Entre los más comunes se encuentra la implicación insuficiente del cliente, ya que estos no comprenden la importancia de trabajar con rigor en la obtención de los requisitos para garantizar la calidad de los resultados, trayendo consigo a largo plazo problemas en la validación del producto obtenido.

Otro de los defectos son los requisitos crecientes y cambiantes. Esto puede incrementar o modificar funcionalidades ya implementadas, desbordando costos y agendas planificadas, generándose también parches de programación que pueden trastocar principios básicos de diseño y degradar la arquitectura del sistema obteniéndose finalmente un producto con serias deficiencias técnicas.

La ambigüedad es una falla habitual de las descripciones de requisitos. Esta crea expectativas diferentes entre las partes del proyecto, y hace que los desarrolladores programen funcionalidades que no se ajustan

a lo que los usuarios necesitan (re-programación), ocasionando la pérdida de tiempo en re-codificación, influyendo negativamente en el tiempo de entrega del producto al cliente final.

Es frecuente la tendencia de algunos desarrolladores a incluir funcionalidades que no figuran en la especificación de requisitos, suponiendo que los usuarios lo agradecerán y que en su mayoría quedan programadas pero sin uso, suponiendo un coste de desarrollo innecesario. También es frecuente que el cliente pida funcionalidades que en realidad no añaden funcionalidad al producto, suponiendo un esfuerzo importante de desarrollo.

En ocasiones el cliente tiene tan sólo el concepto general del producto que desea. La tentación en estos casos es partir de una descripción mínima e ir preguntando y revisando a los programadores conforme el desarrollo avanza. Las estimaciones prematuras, basadas en información limitada pueden fácilmente desbordarse en más del doble. Siempre que sea preciso ofrecer valoraciones previas es conveniente ofrecer varias posibilidades, o incluir un porcentaje posible de error probable. [13]

Importancia de los requisitos en el desarrollo del software

Pressman expresa que para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. [Pressman, 1995] [17]

La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan ardua como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si se realiza de forma errónea. Ninguna otra parte es tan difícil de rectificar posteriormente. [Frederick P. Brooks, 1995] [18]

Luego de analizar lo planteado por los autores anteriores se puede afirmar que la calidad con que se realice la captura de los requisitos va a influenciar en todo el proceso de desarrollo del software repercutiendo en el resto de las fases de desarrollo del mismo. Una definición eficiente de los requisitos permite mostrar un nivel de disciplina en el proceso de desarrollo, dar un mejor soporte a la Gestión de Cambios y ganar una mayor eficiencia en las pruebas reduciendo el riesgo, mejorando la calidad y permitiendo la automatización. Además contribuye a tomar mejores decisiones de diseño y de arquitectura. También le permite al equipo de desarrollo reducir los problemas de mantenimiento.

Los errores en los requisitos se comportan como una enfermedad contagiosa que repercute en todas las fases del proyecto. En el anexo 2 se muestra una tabla donde se argumenta lo anteriormente plantado. [13]

1.4.3 Requisitos no funcionales

El riesgo de no poder terminar el producto en tiempo y forma, creciendo la posible consecuencia del abandono del mismo, es un escenario común que se mantiene en muchos proyectos informáticos hasta el día de hoy. Una de las causas que lo provoca es que los requisitos no funcionales no fueron especificados en el tiempo, o estudiados con la atención explícita de los riesgos que involucran, dificultando así el tratamiento de los problemas de calidad que puedan surgir. La identificación y gestión de los riegos asociados a los requisitos no funcionales desde la fase de Ingeniería de Requisitos puede permitir minimizarlos, evadirlos y controlarlos. Reconociendo los riegos técnicos en un proyecto se pueden identificar los requisitos no funcionales del mismo.

Un **riesgo** es aquel factor que influye negativamente en el éxito del proyecto. RUP lo define como una variable que dentro de su distribución normal puede tomar un valor que pone en riesgo o puede eliminar el éxito del proyecto. En términos sencillos es todo aquello que puede interponerse en el camino hacia el éxito y que es desconocido o incierto. [15]

Pressman considera tres categorías de riesgos: *riegos del proyecto* (amenaza al plan de proyecto), *riesgos técnicos* (amenazan la calidad y la planificación temporal del software) y *riegos del negocio* (amenazan la viabilidad del software). [Pressman, 2002] En el presente trabajo se hará énfasis en los riesgos técnicos ya que estos repercuten principalmente en los requisitos no funcionales y en los requisitos de calidad.

Pressman expresa que si un riesgo técnico se convierte en realidad la implementación puede llegar a ser difícil o imposible. Los riesgos técnicos identifican problemas potenciales de diseño, implementación, de interfaz, de verificación y de mantenimiento. [Pressman, 2002] [19]

Es de vital importancia que se realice una identificación de los riesgos con total calidad, que se le dedique tiempo a esta actividad ya que así se evitarían más trastornos durante el proyecto. El enfrentamiento proactivo de los riesgos que pueden afectar al desarrollo o a la calidad de los requisitos y las acciones para evitarlos, permitirían minimizar problemas que persisten en el desarrollo de software. Son de mayor importancia los riesgos asociados a las principales características de calidad de los requisitos. Por lo que

se concluye que para llegar a obtener una buena captura de requisitos, entre otras cosas, es necesario considerar todos los posibles riesgos que puedan aparecer en el futuro que afecten la calidad del proyecto en general.

La identificación, comprensión y la importancia de la correcta y precisa especificación de los requisitos, se incrementa drásticamente con la magnitud y complejidad de los sistemas de software, particularmente si estos tienen que responder a exigencias que van más allá del alcance de sus funcionalidades principales. Estas exigencias deben ser identificadas temprano y consideradas oportunamente por los profesionales del software en el desarrollo de sus aplicaciones y han enfocado la atención de la comunidad de investigadores hacia la disciplina de la Ingeniería de Requisitos, en particular hacia una Ingeniería de Requisitos de Calidad, en la cual se destaca explícitamente el tratamiento dado a los **requisitos no funcionales**.

Los requisitos no funcionales como su nombre sugiere, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. [Sommerville, 2005].

Sommerville expresa que el incumplimiento de un requisito no funcional puede significar que el sistema entero sea inutilizable. Otras de las dificultades asociadas a estos requisitos es que no hay reglas ni lineamientos para determinar cuándo se obtuvo una solución óptima. Además deben expresarse de forma tal que puedan ser verificados. Los requisitos no funcionales pueden ser más críticos que los funcionales, ya que son estos los que influyen directamente en la arquitectura, y si estos no son correctos el software puede no funcionar o no cumplir con las expectativas del cliente.

Clasificación de los requisitos no funcionales

Según el profesor Sommerville los requisitos no funcionales pueden venir de las características requeridas del software, de la organización que desarrolla el software o de fuentes externas. Ver figura 9.

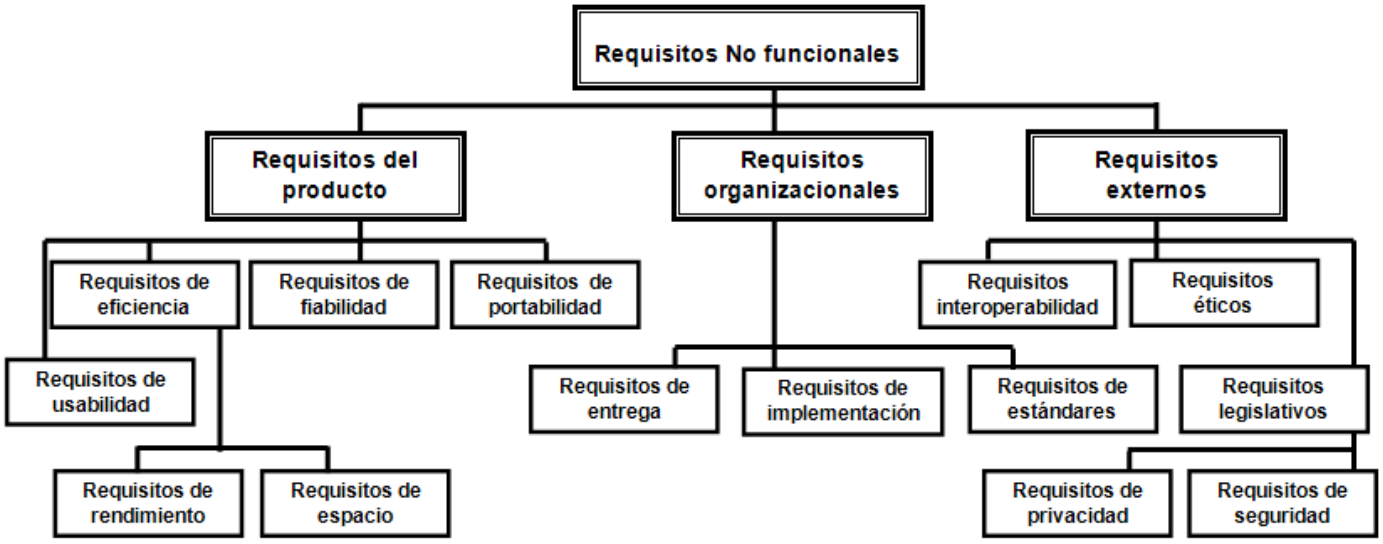

Figura 9. Clasificación de los requisitos no funcionales según Sommerville [2005].

Como se puede observar en dicha figura (figura 9) el autor ignora los requisitos de calidad dentro de los requisitos no funcionales. Mientras que el autor Juan C. Fernández Galante, propone tres áreas dentro de los requisitos no funcionales: *negocio*, *técnico* y *de calidad*. [20]

- **Requisitos de negocio:** Para los negocios, no hay razones técnicas. Por ejemplo, "A fin de ampliar la base de clientes potenciales, se debe interactuar con las herramientas de XYZ." La mayoría de estos requisitos también son no negociables.
- **Requisitos técnicos:** Se limitan las opciones de diseño mediante la especificación de algunas tecnologías que se deben utilizar. Estos requisitos son por lo general, no negociables.
- **Requisitos de calidad:** Definir los requisitos de una aplicación en términos de escalabilidad, disponibilidad, facilidad de cambio, la portabilidad, facilidad de uso, por rendimiento.

El doctor Suryn desglosa los requisitos en funcionales y no funcionales como se muestra en la figura 10. Es muy importante observar que a los requisitos no funcionales los divide en requisitos técnicos y requisitos de calidad, punto en el cual coincide con el autor anterior. [21]

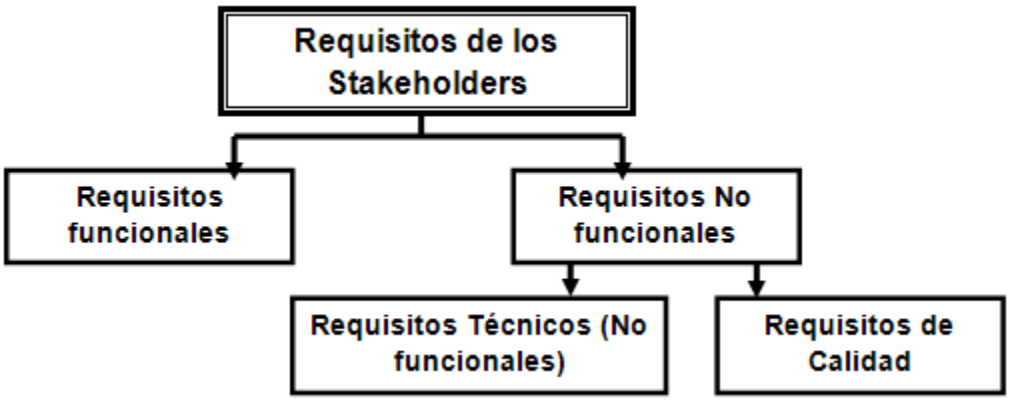

Figura 10. Clasificación de los requisitos según Suryn.

Suryn define una nueva disciplina dentro de la Ingeniería de Requisitos: la Ingeniería de Calidad de Software. Él expresa que el uso de un acercamiento continuo, sistémico, disciplinado, cuantificable al desarrollo y mantenimiento de calidad a lo largo del ciclo de vida de productos de software y sistemas es el uso de la Ingeniería de Calidad de Software. [6] Este es uno de los autores que ha dado los primeros pasos en esta área de requisitos de calidad aún desconocida para muchos que ignoran la importancia de los mismos dentro del desarrollo de un software.

1.4.4 Requisitos de Calidad

Analizando las clasificaciones anteriores se pudo observar que varios autores concuerdan con que los requisitos de calidad figuran dentro de los requisitos no funcionales. Los requisitos de calidad indican cuán bien se está realizando el software. Identificar y entender que atributos de calidad se deben tener en cuenta a la hora de realizar un producto constituye un punto fundamental para que el producto tenga buena demanda.

La temprana obtención de requisitos en un proceso de desarrollo de software permite mejorar la calidad del producto. La calidad es un asunto de competitividad, esencial para sobrevivir y exportar. Es rentable, retiene clientes y aumenta las ganancias. Por todo lo anterior se debe prestar mucha atención a la hora de definir los requisitos de este tipo que además son lo más difíciles de especificar. Se necesitan requisitos de calidad que establezcan desde un principio lo que se espera de un producto final. Es absolutamente necesario que se trabaje en identificar los factores de calidad relevantes para cada tipo de producto a desarrollar. [22]

En el caso específico de los métodos para la identificación de requisitos de calidad, éstos son escasos. Adicionalmente, las organizaciones no tratan de manera específica la captura de requisitos de calidad. Este tipo de requisito se considera como cualquier otro requisito excluyéndolos o mezclándolos con los requisitos funcionales y no funcionales.

A pesar de no existir ninguna definición de requisitos de calidad, se concluye al final de todo el estudio de la bibliografía, que un requisito de calidad es una característica de calidad que afecta en mayor o menor medida el software que se está desarrollando. Es decir, un requisito de calidad va más allá de las características y subcaracterísticas de calidad definidas anteriormente. Estos son mucho más específicos.

Los requisitos de calidad afectarán en mayor o menor medida en dependencia del software que se quiera desarrollar. Ejemplo de esto sería el caso de un software para un cajero electrónico. Aquí la subcaracterística de calidad seguridad juega un papel decisivo. Dentro de esta subcaracterística existirían varios requisitos de calidad como: seguridad de la base de datos, seguridad al autenticarse, seguridad de las transacciones. Si se deja de tener en cuenta un solo requisito de calidad el producto final no tendrá el éxito deseado. Pero si se quiere desarrollar una aplicación para el centro de genética médica, la seguridad de los datos seguiría siendo una subcaracterística importante, pero no afectaría tanto como es el caso de la subcaracterística exactitud de los resultados que ahí se manejan.

1.5 Situación existente en la Universidad de las Ciencias Informáticas

Hoy en día en la Universidad de las Ciencias Informáticas se desarrollan innumerables proyectos de uso nacional y de exportación, lo cual ha traído como consecuencia que cada día se incrementen las investigaciones sobre la calidad de estos proyectos y aunque se le realizan constantemente pruebas y revisiones para controlar la calidad con la que se van desarrollando, muchos no salen en tiempo y no cumplen adecuadamente con las expectativas de los clientes. Por la magnitud que han alcanzado los proyectos en la universidad, el peso que tienen dentro de las aspiraciones de la UCI y el impacto económico-social que puede representar para el país tanto por la venta nacional como la exportación de software, garantizar la calidad del software es una necesidad.

En los últimos años, se han introducido importantes mejoras en el desarrollo de software, especialmente en las etapas de Gestión de proyectos, Codificación y Pruebas. Pero la mayor fuente de defectos sigue siendo los Requisitos. Una de las principales causas es la falta de conocimiento en la captura de requisitos, en específico, la ausencia de métodos para la identificación de requisitos de calidad. Los

proyectos no tratan de manera específica la identificación, especificación y validación de los requisitos de calidad, y en la mayoría de los casos, lo ignoran. Ni estudiantes ni profesionales pertenecientes a los proyectos productivos tienen noción de estos requisitos, desconociendo además la importancia de los mismos.

El departamento de Calidad cuenta con un sitio el cual pone a disposición de todos los proyectos planillas y herramientas para mejorar la calidad de los productos que se producen en el centro. Pero en este sitio no aparece ningún documento donde guíe o explique la forma adecuada de capturar, describir y validar los requisitos incluyendo los de calidad. Es conocimiento de todos que los errores en los requisitos son muy caros de corregir una vez desarrollado el sistema y aún así no se ha construido un método o procedimiento que muestre como identificar los requisitos de calidad en la universidad.

1.6 Conclusiones del capítulo

De acuerdo a las definiciones vistas, la calidad de un sistema o producto de software está estrechamente relacionada con la calidad con que se eliciten los requisitos. Pero en ocasiones solo se le da importancia a los requisitos funcionales, ignorando el hecho de que estos solo representan un subconjunto de las expectativas que deben ser alcanzadas para que el producto final cumpla con las necesidades del cliente.

En este capítulo se especificaron conceptos relacionados a la calidad de software, haciendo mención de normas y estándares de calidad. Se fundamentó sobre los temas de Ingeniería de Requisitos incluyendo la Especificación de Requisitos. Se analizaron diferentes puntos de vista de varias personalidades dentro del ámbito de los requisitos y la calidad de software.

En todo el desarrollo del capítulo se destacó la importancia de la fase de Requisitos dentro de un proyecto, en específico, la importancia de los requisitos no funcionales y dentro de estos los requisitos de calidad. Se evidenció la necesidad existente en la universidad de un procedimiento que guíe el proceso de identificar y especificar este tipo de requisito.

CAPÍTULO II: SOLUCIÓN PROPUESTA

2.1 Introducción del capítulo

La captura de requisitos es la actividad considerada como el primer paso en un proceso de Ingeniería de Requisitos. Pero se ha demostrado que se presentan problemas en esta fase y además se ignoran requisitos de vital importancia para el software. En este capítulo se presenta la propuesta de un procedimiento que les permitirá a desarrolladores de los distintos proyectos de la universidad identificar y especificar los requisitos de calidad dentro de su proyecto, mejorando aún más la calidad del producto final.

2.2 Descripción general del procedimiento

Este procedimiento propone una vía para identificar y especificar los requisitos de calidad luego de haberse obtenido los posibles riesgos técnicos que afectarían al producto. Para aplicar dicho procedimiento no hace falta el uso de ninguna herramienta informática complicada, solo se hace uso del Excel y el Word, y se puede trabajar en cualquier sistema operativo.

El procedimiento se aplicará inicialmente en la fase de Requisitos, pero será un proceso iterativo e incremental, por lo que se podrá aplicar en el resto de las fases en caso de que se haga algún cambio ya sea en la misma fase de requisitos o en el resto de las fases. Se repetirá el procedimiento tantas veces se produzca un cambio con el objetivo de identificar nuevos requisitos de calidad.

Objetivo: El objetivo de este procedimiento es guiar a desarrolladores de un proyecto en el proceso de identificar y especificar requisitos de calidad. Este procedimiento les permite realizar estas actividades correctamente.

Alcance: Este procedimiento es flexible ya que es de aplicación en la preparación y realización de todo tipo de proyecto productivo en la universidad siempre y cuando no se altere el orden de las actividades y se lleve a cabo utilizando las planillas y documentos que propone dicho procedimiento.

2.2.3 Panorama de las fases del procedimiento

El procedimiento propuesto contempla dos fases (actividades y pasos a seguir)*:* identificación *y* priorización y especificación de requisitos de calidad, como se muestra en la figura 11. Inicialmente se identifican los requisitos de calidad priorizando los más importantes y finalmente se realiza la

especificación de los mismos. En la descripción de cada una de estas actividades se amplían los principales elementos que la conforman.

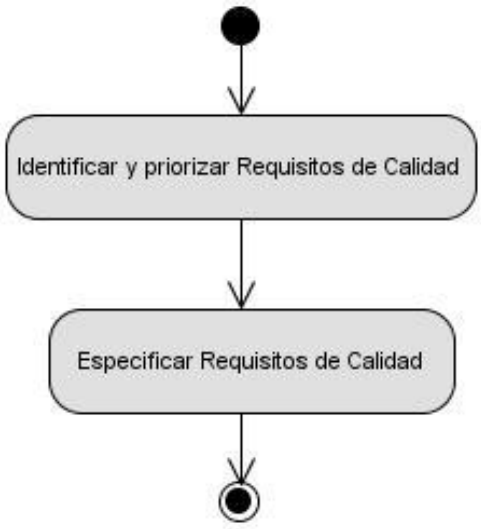

Figura 11. Diagrama del procedimiento propuesto.

Identificar y priorizar requisitos de calidad: En esta fase del procedimiento se propondrá una forma de identificar los requisitos de calidad con el apoyo de los riesgos técnicos que se hayan seleccionado del proyecto. Para la identificación de los requisitos de calidad se hará uso de una lista de chequeo. Luego que estén seleccionados los requisitos se priorizaran en dependencia de cómo los afectan los riesgos técnicos. Esta fase tiene como entrada una lista con los riesgos técnicos del proyecto y una lista de chequeo que servirá para identificar los requisitos de calidad, y como salida una lista con los requisitos de calidad identificados y priorizados.

Especificar requisitos de calidad: En la última fase del procedimiento se propondrá una mejor forma de especificar los requisitos de calidad. Esta fase tendrá como entrada la lista con los requisitos identificados y priorizados y como salida el documento Especificación de requisitos de calidad. En este artefacto se propone una serie de puntos que contribuyen a una mejor especificación de estos requisitos.

En la figura 12 se muestran los artefactos que se generarán en cada una de las fases del procedimiento indicando cuales son los de entrada y los de salida.

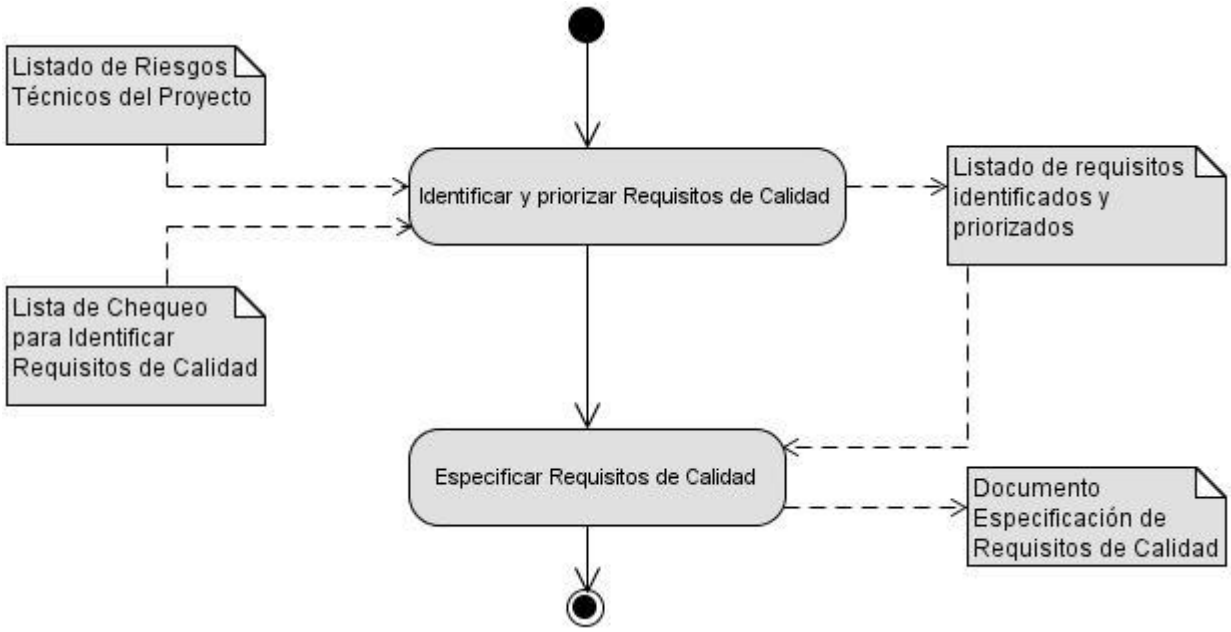

Figura 12. Fases del procedimiento y artefactos generados en cada una de ellas.

2.2.4 Roles y responsabilidades

En el procedimiento intervienen tres roles principales: Especificador de requisitos o Analista, Asegurador de la calidad y el Arquitecto. En la siguiente tabla (tabla 2) se muestra cada uno de los roles con sus responsabilidades y las fases en las que intervienen.

Fases del procedimiento	Rol	Responsabilidad
Identificar y priorizar requisitos de calidad	Especificador de requisitos o Analista Asegurador de calidad Arquitecto	• Identificar los requisitos de calidad en dependencia de los riegos técnicos. • Apoyar el trabajo del especificador. • Priorizar los requisitos de calidad más importantes en dependencia

		a los riegos que más afecten.
Especificar requisitos	Especificador de requisitos o Analista Asegurador de calidad	• Especificar los requisitos que se priorizaron en la fase anterior. • Apoyar el trabajo del especificador.

Tabla 2. Roles y responsabilidades del procedimiento.

2.3 Identificación y priorización de requisitos de calidad

Para lograr producir un software con calidad se necesita identificar los riesgos asociados a los requisitos, de forma que se contribuya al mejoramiento gradual del proceso de desarrollo y la gestión de un proyecto de software que logre la satisfacción del cliente. Para la realización de este procedimiento se necesita inicialmente un listado con los posibles riesgos técnicos que afectarán al producto. La calidad con que se realice esta etapa y las posteriores a ella depende de la buena selección e identificación de los riegos técnicos que se realicen. Aunque es importante destacar que no es objetivo de este trabajo de diploma la identificación de los riegos técnicos.

El SEI propone un método para la identificación de riegos con el objetivo de incrementar las probabilidades de éxito del proyecto. En este método utiliza una taxonomía de los riesgos en el desarrollo de un software (figura 13). La taxonomía inicialmente está organizada en tres clases principales: Ingeniería del producto, Ambiente de desarrollo y Restricciones del programa. [SEI] [23]

- **Ingeniería del producto**: Los aspectos técnicos del trabajo a tener en cuenta para ser logrado.
- **Ambiente de Desarrollo**: Los métodos, procedimientos y herramientas usadas para desarrollar el producto.
- **Restricciones del programa**. Los factores contractuales, de organización, y operacionales dentro de los cuales el software es desarrollado, pero que es generalmente fuera del control directo de la dirección local. [SEI]

Esta taxonomía proporciona un marco para estudiar y organizar los diferentes aspectos del desarrollo del software. En el esquema que se muestra a continuación (figura 13) se puede observar la taxonomía de

clases, descrita desde la perspectiva de los riesgos en el desarrollo de software, la cual se divide en elementos que a su vez son caracterizados por un conjunto de atributos.

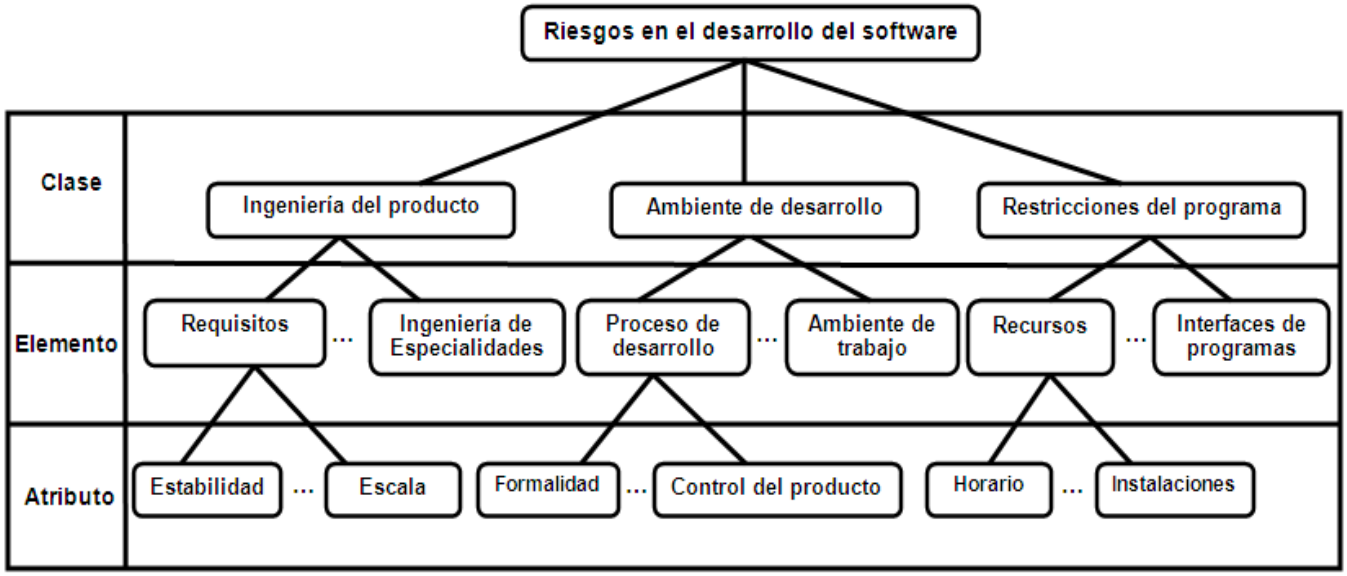

Figura 13. Taxonomía de riesgos en el desarrollo del software según el SEI.

Para el desarrollo de este trabajo se centrará la atención en la clase *Ingeniería del producto* la cual contempla elementos que incluyen atributos asociados a características de calidad. Esta clase se compone de las actividades intelectuales y físicas necesarias para construir el producto que se entrega al cliente. Esta incluye el hardware, software y documentación. A continuación aparece una breve explicación de cada uno de los elementos que conforman la clase Ingeniería del producto:

Requisito: Los atributos que componen esta clase son requisitos de calidad que cubren la calidad de la especificación de los mismos.

Diseño: En este caso los atributos que la componen son requisitos de calidad que pudieran afectar la calidad de la factibilidad de las funciones y algoritmos del software.

Código y pruebas unitarias: Incluye requisitos de calidad que podrían afectar la calidad con que se realiza todo el código del sistema así como las pruebas unitarias que se le realicen.

Integración y pruebas: Los atributos que contiene este elemento son requisitos de calidad que podrían afectar la calidad de la integración de las unidades en un sistema de trabajo y la calidad de la validación que se le realiza al producto de software según las necesidades.

Ingeniería de Especialidades: Este elemento incluye atributos que son requisitos de calidad que podrían afectar las actividades de desarrollo que pueden necesitar conocimientos técnicos especializados, tales como la seguridad y fiabilidad. [23]

En la figura 14 aparecen los elementos que componen la clase Ingeniería del producto incluyendo sus atributos.

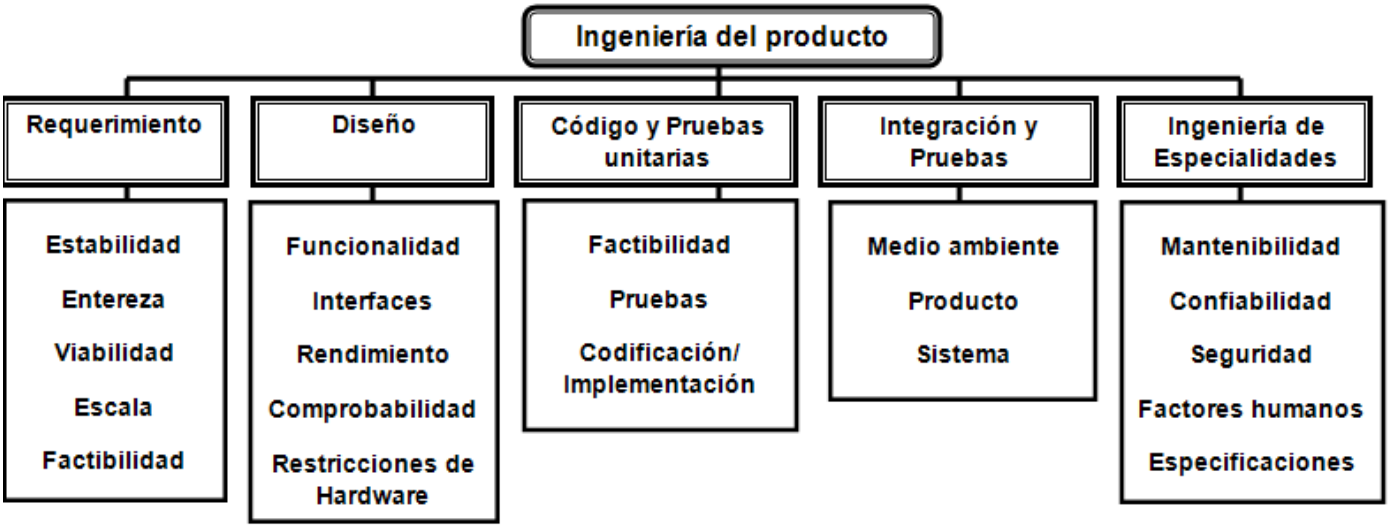

Figura 14. Elementos de la clase Ingeniería del producto según el SEI.

Luego de analizar la definición dada por el SEI se hizo una selección de los atributos que más se relacionan con las características de calidad que se tendrán en cuenta en la realización del presente procedimiento. Se tomaron para esto solo los elementos: Requisito, Diseño e Ingeniería de Especialidades incluyendo algunos de sus atributos.

Las clases Diseño, Código, Integración y Pruebas tienen atributos muy semejantes, por lo que se decidió incluirlos todos en una sola clase llamándola Desarrollo. Por lo que teniendo en cuenta esta clasificación, unida a la clasificación de las características de calidad definidas en el capítulo anterior se concluye que para la realización del procedimiento se tendrán en cuenta las características y subcaracterísticas de calidad que se muestran en la figura 15. Se tomó la división hecha por el SEI para una mejor ubicación a la hora de identificar los requisitos de calidad. Es importante tener claro que cuando se habla de Requisito no se está haciendo referencia al flujo de trabajo definido por RUP. Es una forma de dividir todo el proceso de desarrollo del software.

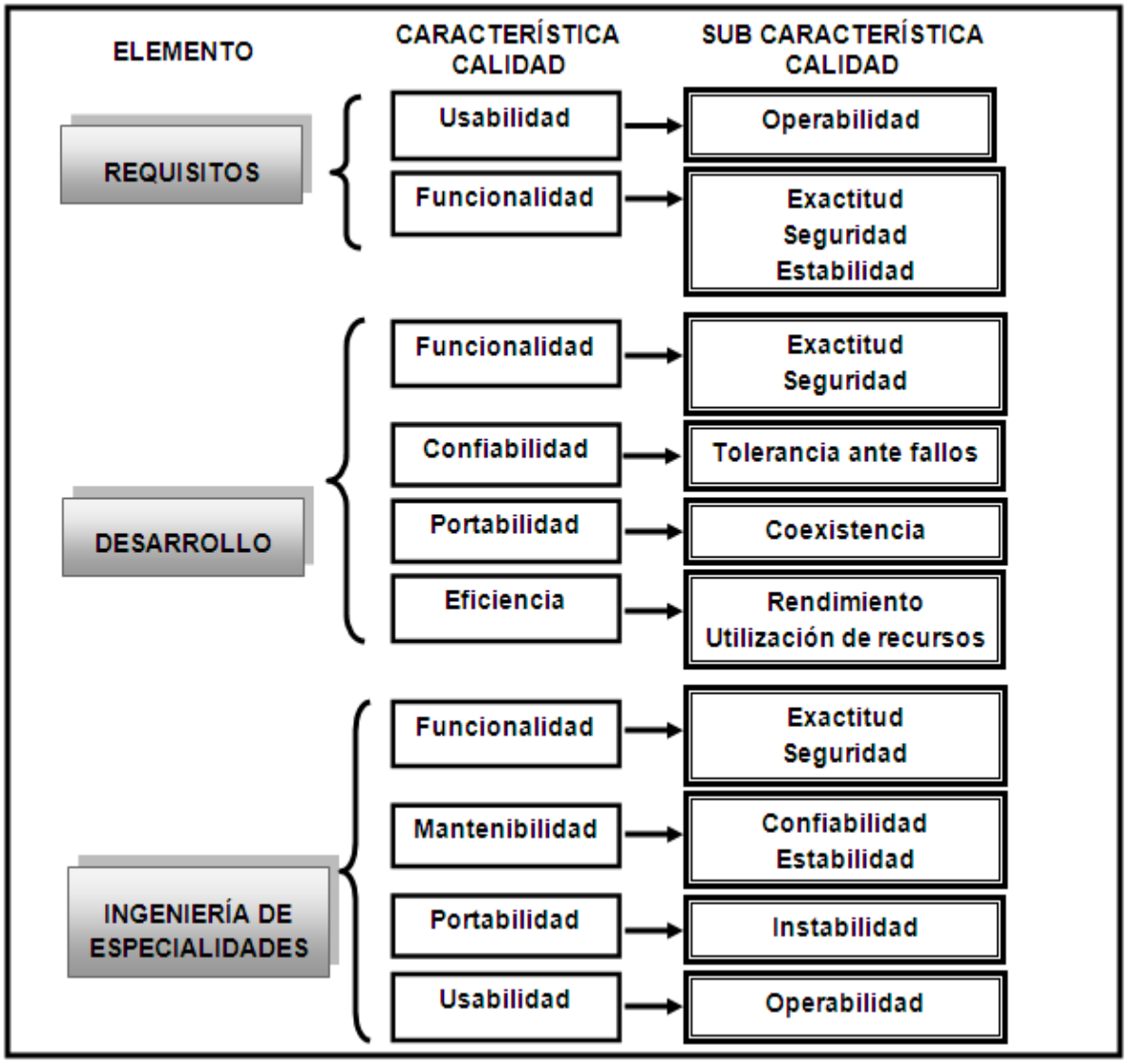

Figura 15. Características y subcaracterísticas de calidad a tener en cuenta en el procedimiento.

La característica de calidad Funcionalidad es de vital importancia por lo que se debe tener en cuenta desde la fase de requisitos hasta la implementación. Se debe seguir la traza hasta al final ya que las subcaracterísticas de calidad que aparecen dentro de ella se le deben dar seguimiento en todo el desarrollo del software. La característica Usabilidad inicialmente aparece en Requisitos ya que es ahí donde se define, y luego aparece en Ingeniería de Especialidades que es donde se materializa.

Artefacto Lista de chequeo para identificar y priorizar requisitos de calidad

Luego de tener claramente definidos los riegos que pueden afectar al desarrollo del producto se hace uso de una lista de chequeo para identificar a partir de estos riesgos qué requisitos de calidad son los que se afectan. La lista de chequeo es considerada como una herramienta de auditoría y es uno de los métodos de evaluación más viejos ampliamente usados.

Este artefacto tiene un carácter flexible para el especificador de requisitos o analista y el especialista de calidad teniendo en cuenta que pueden surgir modificaciones e inclusiones. Es el primer paso para aplicar el procedimiento. La lista de chequeo se le aplicará a todos los desarrolladores del proyecto. Además, pueden participar los clientes para que puedan expresar sus verdaderas necesidades. Esta se desarrollará en formato Excel para que les facilite el trabajo a los especialistas, ya que esta herramienta mediante cálculos y gráficas, les mostrará a simple vista los requisitos de calidad que pueden afectar el software.

La lista que se propone consta de preguntas correspondientes a cada característica y subcaracterísticas de calidad definidas anteriormente (figura 15) para identificar una serie de requisitos de calidad que no se tuvieron en cuenta y que puedan afectar el desarrollo del producto. A continuación aparecen los indicadores a evaluar (*Pregunta con la cual se pretende identificar el requisito de calidad que se ve afectado*) que están en la lista de chequeo. Las mismas están divididas y organizadas por características y subcaracterísticas de calidad. Estos indicadores están escritos de forma clara y de fácil entendimiento para todos aquellos a los que se les aplique la lista de chequeo.

ELEMENTO: REQUISITO		
Característica	**Subcaracterística**	**Indicadores a evaluar**
Funcionalidad	Seguridad	¿Es importante una seguridad extrema en su sistema? ¿Manejará el sistema información confidencial? ¿La pérdida o robo de información puede ocasionar graves pérdidas para la empresa o institución? ¿Afecta al sistema la entrada de intrusos al mismo?

		¿La aplicación se comunicará con otros sistemas para transferir información?
	Exactitud	¿Su sistema trabaja con cálculos? ¿Es de vital importancia el resultado correcto de los cálculos? ¿Un error en los cálculos puede ocasionar graves pérdidas para la empresa o institución? ¿Es importante el grado de precisión de los resultados que se obtengan? ¿Se admite un rango de error en los cálculos que se realicen?
Usabilidad	Operabilidad	¿Es necesario que el sistema sea fácil de operar por el usuario? ¿El sistema está destinado a niños? ¿Los usuarios finales tienen conocimientos informáticos que le permitan trabajar con la aplicación? ¿Qué tan preparados están los usuarios finales para trabajar con la aplicación? ¿Es necesario capacitar a los usuarios del futuro sistema para su utilización?
ELEMENTO: DESARROLLO		
Característica	**Subcaracterística**	**Indicadores a evaluar**
Confiabilidad	Tolerancia ante fallos	¿El sistema debe seguir funcionando si se produce algún fallo? ¿Es importante que el sistema nunca deje de funcionar en el horario previsto?

		¿Es importante que se estructure el sistema de forma tal que se minimicen los daños si falla una parte del mismo? ¿Se puede perder información importante si el sistema deja de funcionar o se pierde la conexión de red? ¿Traería graves consecuencias o pérdidas millonarias para el sistema o empresa un fallo?
Portabilidad	Coexistencia	¿El sistema debe desarrollarse para que funcione en varias plataformas? ¿El sistema necesitará información o datos de otra aplicación para su funcionamiento? ¿El intercambio con otros sistemas es un obstáculo para la seguridad de la aplicación? ¿El sistema debe funcionar en más de un sistema operativo? ¿El sistema va a intercambiar información con otra u otras aplicaciones?
Eficiencia	Rendimiento	¿Es importante que el software de respuestas rápidas? ¿Se realizan transacciones de datos en pequeños instantes de tiempo? ¿Es de vital importancia el tiempo de respuesta del software para el usuario? ¿Se espera que el sistema que se desarrolle se lleve mucho espacio en el disco duro? ¿Se espera que el sistema maneje gran cantidad de información?
	Utilización de	¿El sistema necesitará recursos extras para su

	recursos	funcionamiento? ¿El sistema puede funcionar sin los recursos? ¿Qué cantidad de recursos extras necesita el sistema? ¿Los recursos con los que debe trabajar se consideran de vital importancia para el buen funcionamiento del software? ¿Se espera que sea un sistema distribuido?
ELEMENTO: INGENIERÍA DE ESPECIALIDADES		
Característica	**Subcaracterística**	**Indicadores a evaluar**
Mantenibilidad	Confiabilidad	Si el sistema falla: ___ ¿Es de vital importancia que se hagan salvas? ___ ¿Es importante la consistencia de los datos generados? ___ ¿El sistema genera información que cambia continuamente? ___ ¿Es importante que el software guarde trazas? ___ ¿Es de vital importancia que el sistema guarde los cambios que se estaban realizando antes del fallo?
	Estabilidad	¿Se espera que se le realicen mejoras o cambios al software en un futuro? ¿Se conectarán muchos usuarios al sistema? ¿Se espera que el número de usuarios aumente con el tiempo? ¿El sistema almacenará gran cantidad de información por largo tiempo? ¿La información que guarda el sistema aumentará en

		dependencia que pase el tiempo?
Portabilidad	Instabilidad	¿Es responsabilidad del usuario final instalar el sistema? ¿Debe ser de conocimiento del usuario como instalar el sistema? ¿Se hará un instalador de la aplicación? ¿Es de interés del usuario que la aplicación sea portable? ¿Presenta alguna limitación de hardware que impida el funcionamiento del software?

Tabla 3. Indicadores de la lista de chequeo.

Cada una de las preguntas se evaluará con los valores: alto, medio y bajo. Si una de ellas es valorada como alta significa que es muy importante para el proyecto, y así sucesivamente para los valores medio y bajo. En algunos casos los valores a seleccionar no son de alto, medio o bajo, sino que varían en dependencia a la información que se desea conocer. Aunque en todos los casos se le da un peso a cada una de las variantes a escoger según su importancia. En las siguientes figuras (figuras 16 y 17) se muestran fragmentos de la lista de chequeo en el Excel.

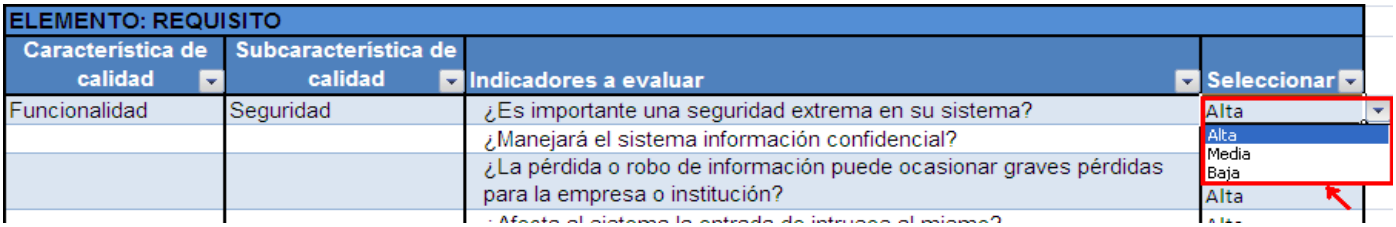

Figura 16. Ejemplo 1 de la Lista de Chequeo en el Excel.

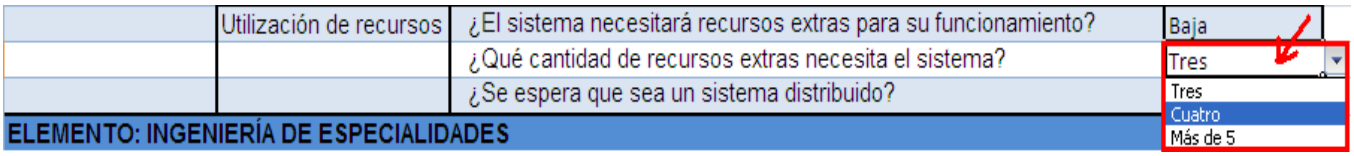

Figura 17. Ejemplo 2 de la Lista de Chequeo en el Excel.

Al final de las preguntas aparece un cuadro resumen donde se muestra el peso que alcanzó cada subcaracterística de calidad que se tuvo en cuenta en la lista de chequeo. El peso se calcula teniendo en cuenta la variante marcada. Al valor de alta se le dio un peso de 5, a medio 4 y a bajo 3. En el caso de las otras variantes también se les dio un peso que indica la importancia de la misma. Para cada subcaracterística se le va sumando el peso de cada respuesta.

Para una mejor comprensión se muestra en el Excel una gráfica donde se puede observar a simple vista la o las subcaracterísticas que van a afectar en mayor medida al software. En la figura 18 se muestra un ejemplo de la gráfica.

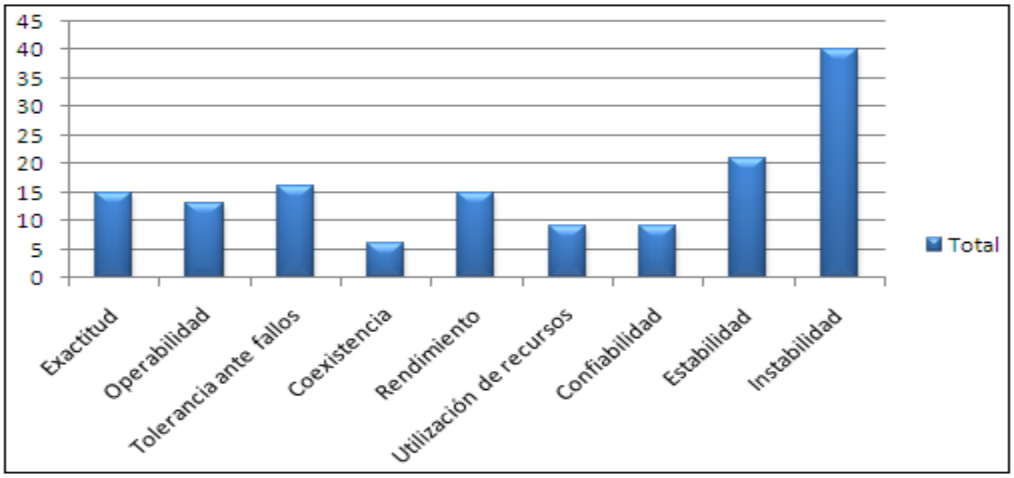

Figura 18. Ejemplo de la gráfica que muestra el Excel.

2.4 Especificación de requisitos de calidad

Después de haber identificado y priorizado los requisitos de calidad según ha indicado el procedimiento, se procederá a la etapa de Especificación de requisitos de calidad, obteniendo como entrada el listado con los requisitos que fueron identificados y priorizados. En esta fase se propondrá una plantilla que guiará al especificador a realizar una buena especificación de cada uno de los requisitos priorizados.

Plantilla de Especificación de Requisitos de Calidad

En este documento la especificación se llevará a cabo a través de diferentes aspectos que ayudarán a entender mejor la aplicación del requisito y cómo influirá en el desarrollo del proyecto. Este artefacto será

de carácter flexible para el especificador de requisitos o analista y el asegurador de la calidad ya que pueden surgir diferentes modificaciones.

Los aspectos que conforman este artefacto son los siguientes:

- **Característica y subcaracterística de calidad:** Nombre de la característica y subcaracterística de calidad que incluye al requisito de calidad a especificar.
- **Nombre del requisito de calidad:** Nombre del requisito a especificar.
- **Descripción:** Descripción general del requisito de calidad.
- **Sobre que artefacto del sistema influye:** Artefacto del sistema que se afecta (Artefactos: Clase, módulo, Caso de Uso).
- **Sobre que flujo de trabajo influye:** Flujo de trabajo que se ve afectado.
- **Cómo medirlo:** Se describe la métrica o fórmula para realizar esta actividad.
- **Cómo probarlo:** Se describen las pruebas o revisiones realizadas para probar el requisito de calidad en cuestión.

Este artefacto solo indica los puntos que se deben tener en cuenta para una buena especificación. Es responsabilidad del especificador o analista y del asegurador de calidad definir, en dependencia del requisito con el que se está trabajando, como medir y probar estos requisitos mediante métricas u otras vías.

2.5 Conclusiones del capítulo

En todo el desarrollo del capítulo se detalló todo el procedimiento propuesto para identificar y especificar a los requisitos de calidad de cualquier proyecto informático una vez identificados los riesgos técnicos que podrían afectarlo. Se describieron las fases que lo componen explicando los artefactos que se generan en cada una de ellas. La propuesta realizada facilita la identificación y especificación de los requisitos de calidad, disminuyendo los riesgos que afectarían a cualquier proyecto productivo de la universidad.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción del capítulo

El desarrollo de un procedimiento para la identificación y especificación de los requisitos de calidad de un proyecto provee una mayor seguridad a la hora de desarrollar un software pues proporciona una primera base para que el producto final tenga buena demanda por el cliente.

En este capítulo se abordará acerca de la validación y análisis de los resultados obtenidos, estableciendo evidencias documentadas. Se explica todo el proceso de validación mediante un panel de expertos y se hace una valoración de los resultados luego de haber aplicado el procedimiento en varios proyectos desarrollados en la universidad. Se describe todo el proceso de aplicación del procedimiento propuesto permitiendo un mejor entendimiento de las actividades o fases que lo conforman, demostrando su utilidad para ser aplicado en los proyectos productivos de la universidad.

3.2 Validación del procedimiento

Para realizar la validación del procedimiento se escogió el método Delphi. Este consiste en la selección de un grupo de expertos a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, con el objetivo de tratar de conseguir consenso, pero con la máxima autonomía por parte de los participantes. Por lo tanto, la capacidad de predicción de este método se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos.

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.

Este método presenta tres características fundamentales:

- **Anonimato**: Durante un Delphi, ningún experto conoce la identidad de los otros que componen el grupo de debate.
- **Iteración y realimentación controlada**: La iteración se consigue al presentar varias veces el mismo cuestionario. Como se van presentando los resultados obtenidos con los cuestionarios anteriores, se consigue que los expertos vayan conociendo los distintos puntos de vista y puedan ir modificando su opinión si los argumentos presentados les parecen más apropiados que los suyos.

- **Respuesta del grupo en forma estadística**: La información que se presenta a los expertos no es sólo el punto de vista de la mayoría, sino que se presentan todas las opiniones indicando el grado de acuerdo que se ha obtenido. [24]

3.2.1 Fases del método Delphi

Antes de iniciar un Delphi se realizan una serie de tareas previas, como son:

- Delimitar el contexto y el horizonte temporal en el que se desea realizar la previsión sobre el tema en estudio.
- Seleccionar el panel de expertos y conseguir su compromiso de colaboración. Las personas que sean elegidas no sólo deben ser grandes conocedores del tema sobre el que se realiza el estudio, sino que deben presentar una pluralidad en sus planteamientos.
- Explicar a los expertos en qué consiste el método. Con esto se pretende conseguir la obtención de previsiones fiables, pues los expertos van a conocer en todo momento cuál es el objetivo de cada uno de los procesos que requiere el procedimiento. [24]

Fase 1: Formulación del problema

El problema fundamental a resolver es comprobar si el procedimiento que se propone cumple con las expectativas por las cuales fue creado. El objetivo general que se persigue es evaluar mediante la opinión de varios expertos que la propuesta aumenta la calidad con que se realiza la identificación y especificación de los requisitos de calidad de un determinado proyecto.

Fase 2: Elección de expertos

Esta fase presenta dos dimensiones:

- **Dimensión cualitativa:** Dadas las características del procedimiento que se propone, se seleccionaron especialistas de calidad que se desempeñan en diferentes responsabilidades en los proyectos productivos de la universidad y dominen las siguientes temáticas: Procesos de desarrollo de Software, Ingeniería de Software, Ingeniería de Requisitos y Estándares Internacionales de Calidad.
- **Dimensión cuantitativa:** Debido a la falta de profesionales expertos en las esferas necesarias para la validación y al poco tiempo para realizar dicha actividad, se tomó como muestra siete

expertos de calidad, que es el mínimo requerido por el método, en función del tiempo limitado para el desarrollo de la investigación. [25]

Cálculo del coeficiente por competencia

Para la selección de los expertos es de gran utilidad emplear la valoración por competencias. Este método consiste en calcular el coeficiente de competencia (k) del experto a partir de la autovaloración del mismo 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 de la encuesta realizada a los expertos 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 usted	0.3	0.2	0.1
2	Experiencia	0.5	0.4	0.2
3	Trabajo de autores nacionales	0.05	0.05	0.05
4	Trabajo 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 4. Coeficientes de argumentación.

En la tabla anterior (tabla 4) 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. El código de interpretación de los coeficientes de competencias se analiza de la siguiente forma:

- 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 encuentren en nivel bajo no pertenecerán al panel. Los expertos escogidos deben poseer coeficiente de competencia altos y medios. A los expertos que estuvieron de acuerdo a validar el procedimiento se les aplicó los cálculos del coeficiente de conocimiento y por los resultados obtenidos se consideran aptos para emitir un criterio efectivo y decisivo sobre el tema. En el anexo 3 aparece una tabla donde se muestra la caracterización de dichos expertos.

Fase 3: Elaboración y lanzamiento de los cuestionarios

Luego de haber seleccionado los expertos calculando su coeficiente de conocimiento se elaboró un cuestionario para validar la propuesta del procedimiento. La encuesta se llevó a cabo de una manera anónima para evitar influenciar en la opinión de los expertos. En el anexo 4 aparece el cuestionario realizado.

Fase 4: Explotación de resultados

El objetivo de los cuestionarios sucesivos es disminuir la dispersión y precisar la opinión media consensuada. En este caso no fue necesario realizar más de una iteración en la aplicación de los cuestionarios porque las respuestas coincidieron en más de un 95 % a favor de la aprobación del procedimiento propuesto. [25]

3.2.2 Análisis de la encuesta realizada a los expertos

Para una mejor comprensión se realizará un análisis de las preguntas que conformaron la entrevista hecha a los expertos. Los resultados arrojados por la dicha encuesta fueron los siguientes:

Analizando las dos primeras preguntas que responden a la importancia de la realización y aplicación del procedimiento, se realizó un gráfico (figura 19) que muestra la opinión dada por los expertos.

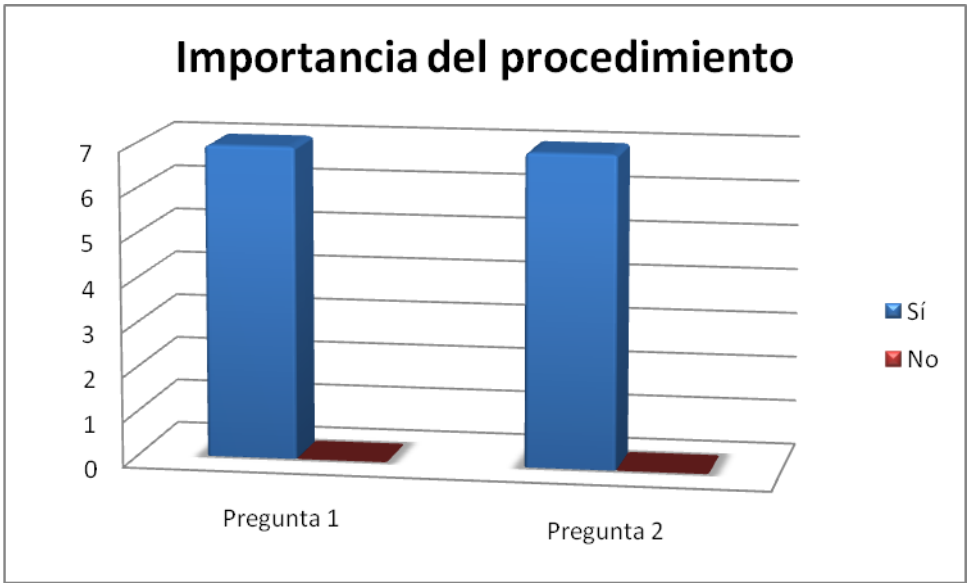

Figura 19. Opinión de los expertos sobre la importancia del procedimiento.

En la siguiente figura (figura 20) se muestra la valoración de los expertos acerca de los siguientes criterios:

- Necesidad del empleo de la propuesta.
- Posibilidad de aplicación.
- Aporte en el proceso de definición de requisitos de calidad en un proyecto.
- Impacto en la Calidad de los Requisitos.

Solo dos expertos expresaron que el procedimiento no tenía mucha posibilidad de ser aplicado, el resto reflejó que si le veían posibilidad de ser aplicado a los proyectos productivos de la universidad.

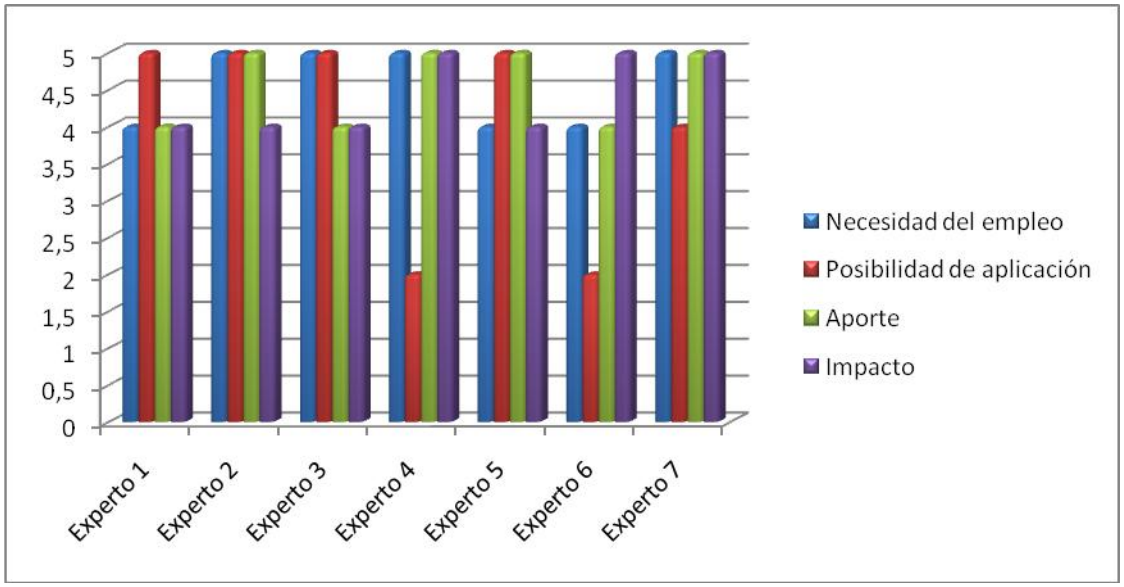

Figura 20. Opinión de los expertos sobre la necesidad, aporte e impacto del procedimiento

La opinión de los expertos en relación a la calidad del procedimiento se muestra en la siguiente tabla (tabla 5):

Preguntas	Opinión de los expertos		
	Innecesarias	**Necesarias pero no suficientes**	**Necesarias y suficientes**
Las actividades propuestas son:	0	1 (15 %)	6 (85%)
Los artefactos propuestos son:	0	2 (28%)	5 (72%)
La asignación de roles y responsabilidades por fases del procedimiento es:	0	1 (15 %)	6 (85%)

Tabla 5. Opinión de los expertos sobre la calidad del procedimiento.

Como se puede observar la mayoría de las valoraciones dadas por los expertos son positivas para el procedimiento. En su totalidad estuvieron de acuerdo con que la organización de las actividades, los roles definidos y los artefactos generados del procedimiento que se propone son correctos. En cuanto a la

elección de las características y subcaracterísticas de calidad realizada, todos estuvieron de acuerdo en que es bastante completa.

De forma general todos los expertos dieron muy buenas opiniones sobre el procedimiento, proponiendo que el mismo se aplicara en todos los proyectos de la universidad. Algunos de ellos dieron muy buenas recomendaciones para incrementar y mejorar la calidad del mismo. Todos lo calificaron como muy útil para mejorar la calidad con que se realizan las aplicaciones informáticas en la universidad.

Cabe destacar algunas de las opiniones dadas:

- Considero que la aplicación del procedimiento contribuye a la mejora de la captura de los requisitos de calidad ya que el mismo aumenta el nivel cultural de los especialistas informáticos en estos temas considerados de menor importancia en el desarrollo de aplicaciones y ayuda en el cumplimiento de la disciplina tecnológica de los involucrados en el desarrollo de estas actividades dentro del proyecto.
- Considero que es de mucha ayuda y muy práctico el procedimiento para identificar los atributos de calidad del software debido a que facilita el trabajo del analista ya que brinda una guía de los elementos y pasos que se deben seguir para identificar y especificar los mismos.

3.3 Aplicación del procedimiento

Para aplicar el procedimiento inicialmente se buscaron varios proyectos ya terminados de la universidad. Cuando ya se tuvieron los proyectos sobre los cuales se aplicaría dicho procedimiento, se procedió a buscar la documentación que reflejaba el levantamiento de requisitos que se les había realizado. Al contactar con los desarrolladores que estuvieron al frente del proyecto, se le realizó una entrevista con el objetivo de que estos explicaran el resultado que se obtuvo luego de instalar completamente el software. Los requisitos no funcionales del proyecto y el resultado final que tuvo la aplicación servirán de comparación una vez que se aplique el procedimiento.

Cuando se habla de resultado final de la aplicación se está haciendo referencia a la aceptación que tuvo con el cliente, si hubo algún problema en la instalación, si al final surgió algún problema relacionado con los requisitos no funcionales que no se previó al inicio o en la etapa de levantamiento de requisitos.

Producto alasEPIGEN

El proyecto EPIGEN desarrolló una aplicación informática llamada alasEpigen que permite realizar análisis estadísticos sobre estudios de Epidemiología Genética. La aplicación desktop desarrollada consiste en realizar estudios epidemiológicos realizando diferentes cálculos estadísticos, dando la opción de escoger el tipo de estudio que se desea realizar y graficar en algunos casos. Este proyecto pertenece a la facultad 6.

Para darle respuesta a la interrogante de cómo había terminado el proyecto seleccionado para aplicarle el procedimiento se utilizó el método de la entrevista. Este método no es más que una técnica de recopilación de datos que se realiza generalmente de forma oral entre dos o más personas. Es una conversación planificada cuyo objetivo es obtener información de uno o varios temas. La entrevista es una técnica eficaz ya que la información que se obtiene es superior que cuando se limita a la lectura de respuestas escritas. A través de estas se pueden captar gestos, tonos de voz que aportan información sobre el tema y las personas entrevistadas. Para procesar dicha entrevista se utilizó solamente el Microsoft Word 2007. En el anexo 5 aparece la entrevista completa hecha a los líderes y desarrolladores.

La entrevista se le realizó a su antiguo líder de proyecto el Ingeniero Yosuán Crespo García, instructor recién graduado con un año y medio de experiencia. Este reflejó que el software no está instalado formalmente en ningún organismo, pero si se le han realizado numerosas pruebas en las cuales presentó algunos problemas vinculados a los requisitos no funcionales. Luego de la entrevista se pasó a la revisión del levantamiento de requisitos que se había realizado y acto conjunto se le hizo entrega de la lista de chequeo.

Luego de aplicar la lista de chequeo al líder de proyecto y estudiar el levantamiento de requisitos no funcionales del proyecto, se observó que en dicho levantamiento no tuvieron en cuenta algunos requisitos de calidad que sobresalieron luego de realizar dicha actividad. En la gráfica que se muestra a continuación (figura 21) se puede observar que la subcaracterística de calidad exactitud tiene gran importancia dentro del proyecto. Y cuando se revisó el documento donde estaban reflejados los requisitos no funcionales, esta subcaracterística no se había tenido en cuenta, o por lo menos, no aparece definida ni especificada.

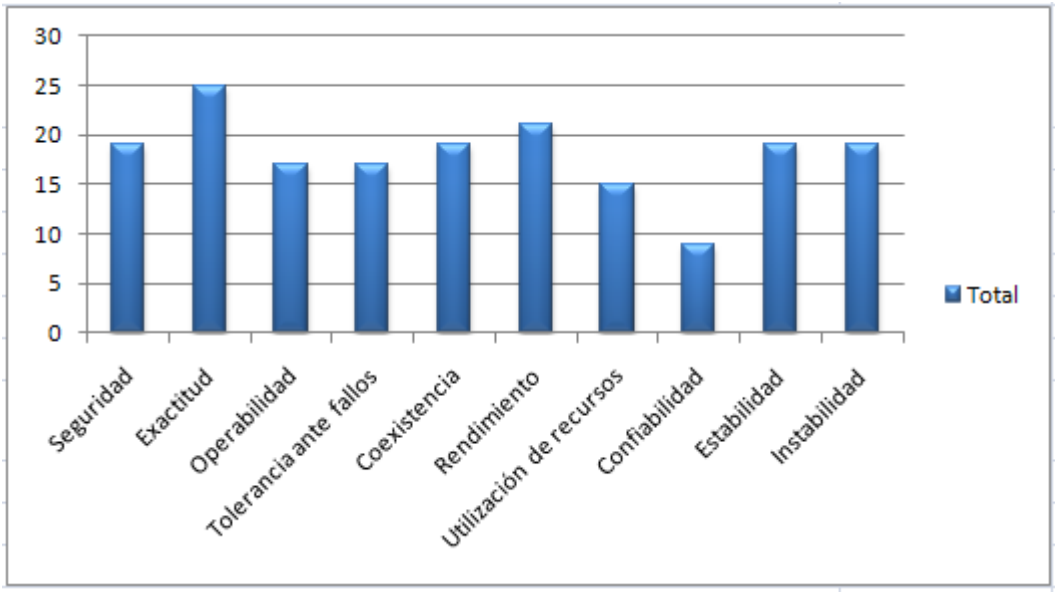

Figura 21. Resultado de la lista de chequeo para alasEPIGEN.

El requisito de calidad exactitud de los cálculos afecta en gran medida al software ya que se trata de una aplicación destinada a estudios epidemiológicos que realiza diferentes cálculos estadísticos. Al no tenerse en cuenta este requisito en el desarrollo del software se han presentando problemas en las pruebas que se le han realizado.

Otro de los requisitos que se considera importante es la seguridad del sistema. El entrevistado reflejó que la aplicación debía manejar información confidencial, pero a pesar de esto, no se tuvo en cuenta en las descripciones de los requisitos no funcionales. Se considera que en este tipo de sistemas la seguridad de la información con la que se trabaja es muy importante ya que puede afectar en gran medida el software y aunque el cliente no haga referencia en esto, los desarrolladores deberían hacer hincapié en ello.

Se considera que el requisito de calidad que más prioridad se le debe dar es la exactitud de los cálculos y se debe considerar además la seguridad de los datos. En la siguiente tabla (tabla 5) se muestra un ejemplo de la especificación del requisito de calidad seguridad de los datos.

Característica (subcaracterística de calidad)	Funcionalidad (Seguridad)
Nombre del requisito de calidad:	Seguridad de los datos

Descripción:	Este requisito es de vital importancia en el desarrollo de este software ya que la información con la que se va a trabajar es confidencial.
Sobre que artefacto del sistema influye:	Plan de Gestión de Riesgos
Sobre que flujo de trabajo influye:	Influye en todos los flujos de trabajo.
Cómo medirlo:	*Los desarrolladores deben definir las métricas que van a utilizar.*
Cómo probarlo:	Pruebas de Cargas y Prueba de stress.

Tabla 6. Ejemplo de especificación de un requisito de calidad

Por todo el análisis realizado se concluye que el procedimiento ha sido efectivo en este proyecto ya que se identificaron nuevos requisitos de calidad que no se tuvieron en cuenta en el levantamiento de requisitos del sistema que se desarrolló. Se recomienda que se tengan en cuenta antes de ser implantado en el sistema de salud.

Proyecto Solución Integral para el perfeccionamiento del sistema de Prevención del Delito de la República Bolivariana de Venezuela

Este proyecto tiene como meta desarrollar un Sistema Informático para la Gestión de Información de las Coordinaciones Regionales adscritas a la Dirección General de Prevención del Delito del Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. El sistema deberá gestionar los datos correspondientes a los usuarios y sus permisos a las funcionalidades del sistema, la información básica de los recursos humanos de las Coordinaciones Regionales, los informes que las Coordinaciones Regionales envían a la Dirección General de Prevención del Delito, entre otras.

En este caso se le realizó la entrevista a la Ingeniera Aidacelys López. La entrevistada explicó que todavía no se había realizado el levantamiento formal de los requisitos ya que el proyecto recién comenzaba pero

a pesar de esto, se interesó en que se le aplicara el procedimiento al proyecto para contribuir a una buena captura de requisitos de calidad.

Al analizar la lista de chequeo se pudo observar que las subcaracterísticas que más podrían afectar al software eran: seguridad, tolerancia ante fallos y estabilidad. En la gráfica (figura 22) que se muestra a continuación se puede observar a simple vista como sobresalen estas subcaracterísticas del resto.

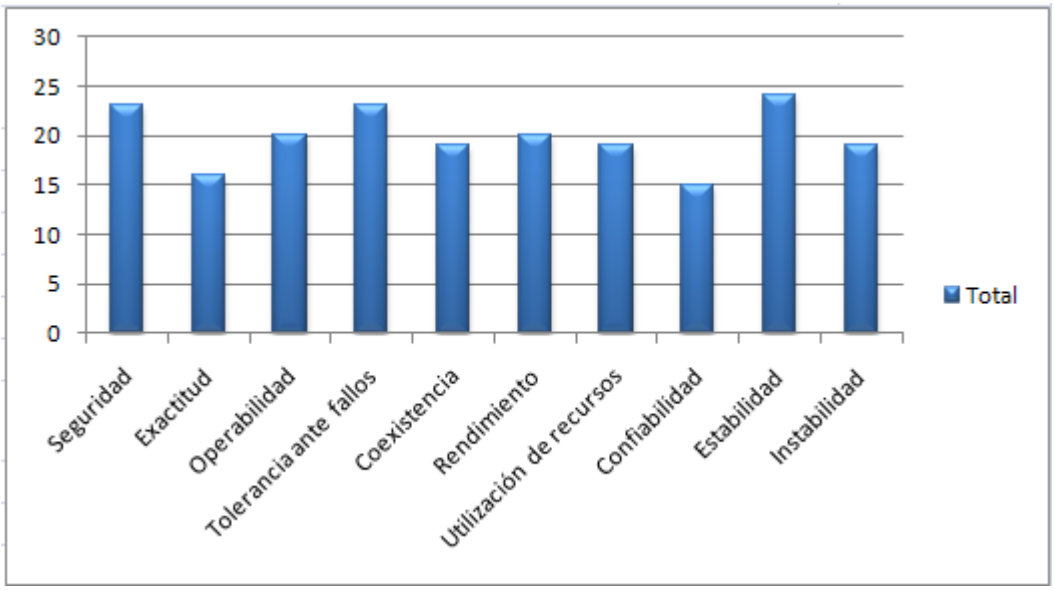

Figura 22. Resultado de la lista de chequeo para el software.

Se recomendó que se tengan muy presentes estos requisitos de calidad a la hora de realizar la captura de requisitos no funcionales del sistema para así poder desarrollar un software con calidad. Teniendo en cuenta que es un software de exportación para Venezuela es muy importante que se realice una correcta identificación de requisitos sin ignorar los requisitos de calidad. Se obtuvieron resultados positivos al aplicar el procedimiento en este proyecto ya que se contribuyó a la calidad de la futura aplicación.

Producto alasMEDIGEN

Otro de los productos a los que se le aplicó el procedimiento fue al Sistema Informático de Genética Médica alasMEDIGEN. Dicho sistema fue desarrollado por el Proyecto de Genética Médica del polo Gestión de Información Biomédica de la Facultad 6. Es una aplicación web que permite gestionar todo tipo de información de pacientes con problemas genéticos, así como todo tipo de discapacidades.

La entrevista en este caso se le realizó a su antiguo líder de proyecto el Ingeniero Yosuán Crespo García. Este expresó que la aplicación no se ha instalado todavía, ya fue aceptada por el cliente y estaban listos para desplegarla. Además reflejó que la única preocupación que tenían era que se presentaran problemas de conexión.

Luego de la entrevista se pasó a la revisión del levantamiento de requisitos realizado por los desarrolladores del proyecto y a la aplicación de la lista de chequeo con el líder de proyecto. Los resultados obtenidos de la lista de chequeo se muestran en la gráfica (figura 23) que se muestra a continuación.

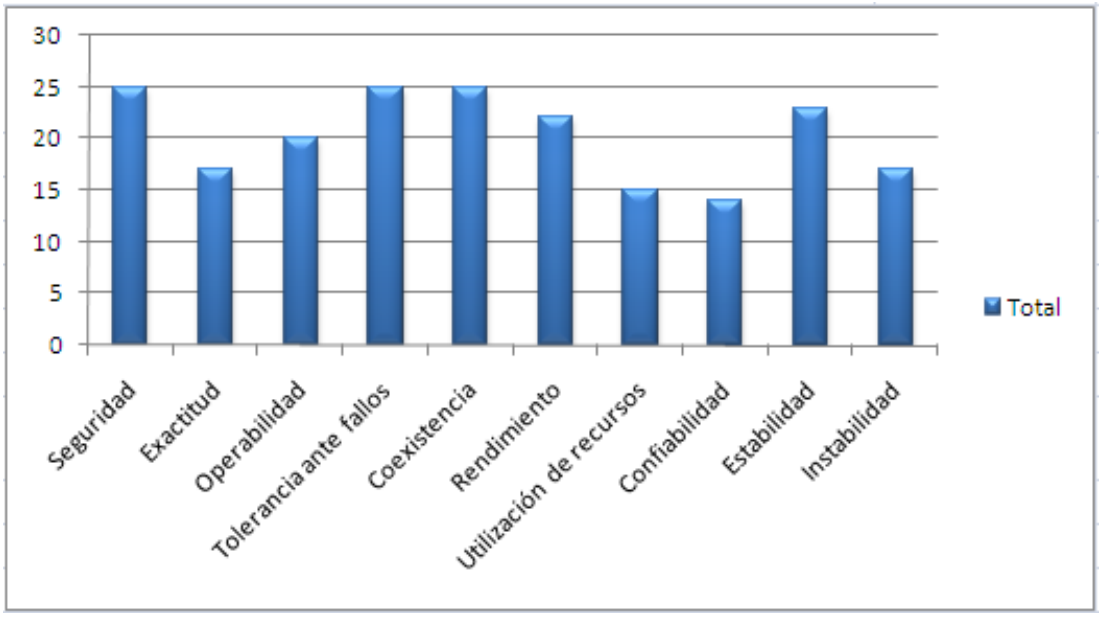

Figura 23. Resultado de la lista de chequeo para alasMEDIGEN.

Al comparar este resultado con los requisitos no funcionales definidos se pudo observar que en el mismo no le daban mucha importancia al requisito seguridad, siendo este uno de los más importantes. El líder de proyecto en la lista de chequeo dejó plasmado que la seguridad era un factor importante ya que se manejaría información confidencial y que la pérdida o robo de los datos con los que trabaja el sistema traería consigo graves consecuencias. Mientras que la descripción de dicho requisito no era muy aclarativa, no se definieron requisitos de calidad tales como seguridad de las base de datos, seguridad de las transacciones de información con otros sistemas, entre otros.

Otra subcaracterística que podría afectar en gran medida al producto es la coexistencia. Una de las preocupaciones del líder de proyecto es la conexión de la aplicación con otros sistemas, sin embargo no especifican en ningún momento la importancia de este requisito de calidad.

Se concluye que el procedimiento ha sido efectivo en este producto ya que se pudo identificar nuevos requisitos de calidad que no se tuvieron en cuenta en el desarrollo del sistema. Se recomienda que se tengan en cuenta antes de ser implantado completamente.

3.4 Conclusiones del capítulo

En este capítulo se ha demostrado la forma de aplicar el procedimiento y los resultados obtenidos luego de haberlo empleado. El análisis de los resultados anteriores permite afirmar que de manera general el procedimiento propuesto es considerado como válido, correcto, completo y seguro para aplicar a los proyectos productivos de la universidad, coincidiendo con los expertos que lo validaron. A su vez es flexible a cambios de acuerdo a las diferentes características que posee el software al cual se aplique, logrando adaptarlo a los fines específicos del mismo.

Luego de haberlo aplicado a dos productos desarrollados en la universidad y a un proyecto que recién inicia se concluye que este procedimiento contribuye a la calidad de las aplicaciones informáticas que se desarrollarán en el centro identificando requisitos que no se tuvieron en cuenta en la realización del software.

CONCLUSIONES GENERALES

Finalizada la investigación se cumplió el objetivo general ya que se obtuvo un procedimiento para identificar y especificar requisitos de calidad dentro de los proyectos productivos de la universidad. A través del estudio realizado se detectaron deficiencias en la captura de requisitos de calidad en algunos proyectos lo cual puede superarse empleando el procedimiento propuesto.

Se aplicó el procedimiento en tres proyectos, dos de ellos ya terminados y otro aún desarrollándose. El resultado que se obtuvo fue satisfactorio demostrándose la efectividad del mismo, ya que se identificaron nuevos requisitos de calidad que no se habían tenido en cuenta y que afectaban en gran medida el correcto funcionamiento del software.

Se validó la propuesta mediante un panel de expertos obteniendo de forma general muy buenas opiniones y recomendaciones para mejoras futuras del procedimiento.

El procedimiento fue definido de manera general y está listo para ser aplicado a cualquier proyecto de la universidad. Los proyectos que apliquen este procedimiento, tendrán un mejor control de los requisitos de calidad que pueden afectarlo en mayor o menor medida en todo el desarrollo del software.

RECOMENDACIONES

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

- Aplicar el procedimiento propuesto a los proyectos productivos que se desarrollen en la universidad.
- Realizar el procedimiento para el resto de las actividades que conforman el proceso de Ingeniería de Requisitos como verificación y validación.
- Desarrollar métricas que permitan medir los requisitos de calidad que sean identificados en la primera fase del procedimiento.

REFERENCIAS BIBLIOGRÁFICAS

1. **Domínguez, Jorge.** THE.PROJECT.MANAGEMENT.HUT. *THE.PROJECT.MANAGEMENT.HUT.* [Online] Junio 16, 2009. [Citado: Noviembre 5, 2009.] Disponible en: http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure

2. **MINREX**. CubaMinrex. Sitio del Ministerio de Relaciones Exteriores de Cuba. *MINREX. CubaMinrex. Sitio del Ministerio de Relaciones Exteriores de Cuba.* [Online] Septiembre 16, 2005. [Citado: Noviembre 7, 2009.] Disponible en : http://www.cubaminrex.cu/Sociedad_Informacion/Cuba_TIC/Informatizaci%C3%B3n.htm

3. ***ISO/IEC 9126.*** *Oficina Nacional de Normalización. Norma Cubana. Ingeniería de Software – Calidad del Producto – parte 1: Modelo de la Calidad (ISO/IEC 9126-1:2001, IDT).* Ciudad de La Habana. Cuba . 2005.

4. ***IEEE 610-1990.*** *Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Computer Dictionary. Software Engineering Terms.* 1990.

5. **Pressman, Roger S.** *Ingenería de Software. Un enfoque práctico.* 1998.

6. **Suryn, Witold.** Software Quality Engineering in international standardization and practice. *Software Quality Engineering in international standardization and practice.* 2007.

7. **Arquitectura de Software. Guía de estudio.** [Online] Abril 2004. [Citado: Enero 14, 2010.] Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guía Arquitectura v.2.pdf

8. **Sommerville, Ian.** *Ingeniería de Software. Séptima edición.* 2005.

9. **Guevara Mojena, Bedsy.** *Procedimiento Propuesto para medir la Calidad en.* Ciudad de La Habana. UCI , 2007. 107 p.

10. ***CMMI****: Guía para la integración de procesos y la mejora de productos. Segunda edición.* Madrid : Cátedra de Mejora de Procesos de Software en el Espacio Iberoamericano de la Universidad Politecnica de Madrid, 2009. 630 p.

11. **Nicolas Ros, Joaquin.** Propuesta de gestión integrada de modelos y requisitos en líneas de productos software. [Online] 2010. [Citado: Febrero 2, 2010.] Disponible en: http://www.tesisenxarxa.net/TESIS_UM/AVAILABLE/TDR-0121110-125950//NicolasRos.pdf

12. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico.* 2006.

13. **Palacio Bañeres, Juan.** *Compendio de Ingeniría de Software I.* 2006.

14. ***IEEE Std 1074-1997****: Standard for Developing Software Life Cycle Processes.* 1997.

15. ***RUP, 2003****.Proceso Unificado de Rational. IBM Rational Suite 2003.* 2003.

16. ***IEEE-STD-830-1998*** *: Recommended Practice for Software Requirements.* 1998.

17. **Pressman, Roger S.** *Ingeniería de Software. : Mc Graw Hill.* 1995.

18. **P. Brooks, Jr. Frederick.** *The Mythical Man-Month. : Addison-Wesley.* 1995.

19. **Pressman, Roger S.** *Ingeniería de Software. : Mc Graw Hill.* 2002.

20. **Fernández Galante, Juan C.** Arquitectura del Software:Requisitos no funcionales. *Arquitectura del Software:Requisitos no funcionales.* [Online] 2010. [Citado: febrero 17, 2010.] Disponible en: http://www.juancarlosfernandez.net

21. **Suryn, Witold.** *Software Quality Engineering.* 2009.

22. **Stockman, Guillaume Sinclair.** "A Framework for software quality Measurement". Volumen 8.

23. **SEI. *Software Engineering Institute.*** Taxonomy -Based Risk Identification**.** Junio 1993.

24. **GTIC: *Grupo de Tecnología de la Información y las Comunicaciones*.** Método Delphi. [Online]. 2010. Disponible en: http://www.gtic.ssr.upm.es/encuestas/delphi.htm#A1.1.2

25. **Eneko Astigarraga**. ***El Método Delphi***. [Online]. 2010. Disponible en: http://www.unalmed.edu.co/~poboyca/documentos/documentos1/documentos-Juan%20Diego/Plnaifi_Cuencas_Pregrado/Sept_29/Metodo_delphi.pdf

BIBLIOGRAFÍA

Arquitectura de Software. Guía de estudio. [Online] Abril 2004. [Citado: Enero 14, 2010.] Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guía Arquitectura v.2.pdf.

***CMMI**: Guía para la integración de procesos y la mejora de productos. Segunda edición.* Madrid : Cátedra de Mejora de Procesos de Software en el Espacio Iberoamericano de la Universidad Politecnica de Madrid, 2009. 630 p.

Calisoft: Centro para la excelencia en el desarrollo de proyectos tecnológicos. [Online] Disponible en: http://calidadsoft.uci.cu

Domínguez, Jorge. THE.PROJECT.MANAGEMENT.HUT. *THE.PROJECT.MANAGEMENT.HUT.* [Online] Junio 16, 2009. [Citado: Noviembre 5, 2009.] Disponible en: http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure.

Eneko Astigarraga. ***El Método Delphi***. [Online]. 2010. Disponible en: http://www.unalmed.edu.co/~poboyca/documentos/documentos1/documentos-Juan%20Diego/Plnaifi_Cuencas_Pregrado/Sept_29/Metodo_delphi.pdf

Fernández Galante, Juan C. Arquitectura del Software:Requisitos no funcionales. *Arquitectura del Software:Requisitos no funcionales.* [Online] 2009. [Citado: febrero 17, 2010.] Disponible en: http://www.juancarlosfernandez.net

GTIC: *Grupo de Tecnología de la Información y las Comunicaciones*. Método Delphi. [Online]. 2010. Disponible en: http://www.gtic.ssr.upm.es/encuestas/delphi.htm#A1.1.2

Guevara Mojena, Bedsy. *Procedimiento Propuesto para medir la Calidad en.* Ciudad de La Habana. UCI , 2007.107 p.

***IEEE 610-1990.** Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Computer Dictionary. Software Engineering Terms.* 1990.

***IEEE Std 1074-1997**: Standard for Developing Software Life Cycle Processes.* 1997.

***IEEE-STD-830-1998** : Recommended Practice for Software Requirements.* 1998.

***ISO/IEC 9126**. Oficina Nacional de Normalización. Norma Cubana. Ingeniería de Software – Calidad del Producto – parte 1: Modelo de la Calidad (ISO/IEC 9126-1:2001, IDT).* Ciudad de La Habana. Cuba , 2005.

MINREX. CubaMinrex. Sitio del Ministerio de Relaciones Exteriores de Cuba. *MINREX. CubaMinrex. Sitio del Ministerio de Relaciones Exteriores de Cuba.* [Online] Septiembre 16, 2005. [Citado: Noviembre 7, 2009.] Disponible en : http://www.cubaminrex.cu/Sociedad_Informacion/Cuba_TIC/Informatizaci%C3%B3n.htm

Nicolas Ros, Joaquin. Propuesta de gestión integrada de modelos y requisitos en líneas de productos software. [Online] 2009. [Citado: Febrero 2, 2010.] Disponible en: http://www.tesisenxarxa.net/TESIS_UM/AVAILABLE/TDR-0121110-125950//NicolasRos.pdf

P. Brooks, Jr. Frederick. *The Mythical Man-Month. : Addison-Wesley.* 1995.

Palacio Bañeres, Juan. *Compendio de Ingeniría de Software I.* 2006.

Plataforma Moddle UCI. [Online]. 2010. Disponible en: http://teleformacion.uci.cu/

Pressman, Roger S. *Ingeniería de Software. : Mc Graw Hill.* 1995.

Pressman, Roger S. *Ingenería de Software. Un enfoque práctico.* 1998.

Pressman, Roger S. *Ingeniería de Software. : Mc Graw Hill.* 2002.

Pressman, Roger S. *Ingeniería de Software. Un enfoque práctico.* 2005.

Pressman, Roger S. *Ingeniería de Software. Un enfoque práctico.* 2006.

RUP*, 2003*.*Proceso Unificado de Rational. IBM Rational Suite 2003.* 2003.

SEI. *Software Engineering Institute.* Taxonomy -Based Risk Identification**.** Junio 1993.

Sommerville, Ian. *Ingeniería de Software. Séptima edición.* 2005.

Stockman, Guillaume Sinclair. “A Framework for software quality Measurement”. Volumen 8.

Suryn, Witold. Software Quality Engineering in international standardization and practice. *Software Quality Engineering in international standardization and practice.* 2007.

Suryn, Witold. *Software Quality Engineering.* 2009.

Tesis.UCI. [Online]. 2010. Disponible en: http://tesis.uci.cu

ANEXOS

Anexo 1: Características y subcaracterísticas de Calidad según la ISO 9126.

Funcionalidad: El grado en que el software satisface las necesidades indicadas por los atributos de calidad
• **Exactitud**: Capacidad del software para proporcionar efectos o resultados correctos o convenidos. • **Seguridad**: La capacidad del software para proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o pueden modificar los mismos, y a las personas o sistemas autorizados no les sea denegado el acceso a ellos. • **Conformidad**: Capacidad del software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares. **** El atributo Conformidad se tiene en cuenta en todos los requisitos de calidad.***
Confiabilidad: Cantidad de tiempo que el software está disponible para su uso.
• **Madurez**: Capacidad del software de evitar una avería como resultado de haberse producido un fallo en el mismo. • **Tolerancia ante fallos:** Capacidad del software de mantener un nivel de ejecución específico en caso de fallos del software o de infracción de sus interfaces especificadas.
Usabilidad: Grado en que el software hace óptimo el uso de los recursos del sistema.
• **Comprensibilidad:** Capacidad del producto para permitirle al usuario entender si el software es conveniente, y cómo puede usarse para las tareas particulares y condiciones de uso. • **Operabilidad:** Capacidad del producto del software para permitirle al usuario aprender de su aplicación.
Eficiencia: Grado en que el software hace óptimo el uso de los recursos del sistema.

• **Rendimiento:** Capacidad del software para proporcionar una respuesta apropiada y los tiempos de procesamiento al realizar su función, bajo condiciones declaradas. • **Utilización de recursos:** Capacidad del software para usar los recursos apropiados en un plazo de tiempo adecuado cuando realiza su función bajo las condiciones declaradas.
Mantenibilidad: Facilidad con que una modificación puede ser realizada.
• **Confiabilidad:** Capacidad del software de operar como es esperado en un intervalo de tiempo especificado. • **Estabilidad:** La capacidad del software para minimizar los efectos inesperados de las modificaciones que se le realizan.
Portabilidad: La facilidad con que el software puede ser llevado de un entorno a otro.
• **Instabilidad:** Capacidad del software de ser instalado en un ambiente específico. • **Coexistencia:** Capacidad del software para coexistir con otro software independiente en un ambiente común que comparte los recursos comunes.

Tabla 7. Características y subcaracterísticas de calidad.

Anexo 2: Consecuencias de los errores en los requisitos en el resto de las fases de desarrollo

Fases	Consecuencias
Estimación	No es posible estimar con rigor costos y recursos necesarios para desarrollar algo que no se conoce.
Planificación	No se puede confiar en la planificación para el desarrollo de algo que no se sabe bien como es.

Diseño	Los errores en requisitos, las modificaciones frecuentes, las deficiencias en restricciones o futuras evoluciones, producen arquitecturas que más tarde se confirmarán como erróneas y serán modificadas.
Construcción	Las deficiencias en los requisitos obligan a programar en ciclos de prueba y error que derrochan horas y paciencia de programación sobre patrones de “recodificación continua” y “programación heroica”.
Validación y verificación	Terminado el desarrollo del sistema, si las especificaciones tienen errores de bulto, o peor aún, no están reflejadas en una especificación de requisitos, no será posible validar el producto con el cliente.

Tabla 8. Consecuencias de los errores de los requisitos en el resto de las fases de desarrollo.

Anexo 3: Caracterización de los expertos escogidos para validar el procedimiento

Expertos	Graduado de:	Categoría Docente	Categoría científica	Labor que realiza	Experiencia
Experto 1	Ingeniería en Ciencias Informáticas	Instructor		Líder de Proyecto	2 años
Experto 2	Ingeniería en Ciencias Informáticas	Instructor		Profesor de Ingeniería de software	3 años
Experto 3	Ingeniería en Ciencias Informáticas	Asistente	Máster	Especialista de CALISOFT	6 años
Experto 4	Ingeniería en Ciencias Informáticas	Auxiliar	Máster	Gestor de Proyecto	14 años

Experto 5	Ingeniería en Ciencias Informáticas	Asistente		Profesor de ISW y Analista de proyecto	6 años impartiendo ISW I y II 2 años Jefa de Desarrollo del SIGESC
Experto 6	Ingeniería en Ciencias Informáticas	Instructor		Asesor de Comercialización del Centro para la Informatización de la Gestión de Entidades (CEIGE) y profesor de la Facultad 15	2 años
Experto 7	Ingeniería en Ciencias Informáticas	Asistente	Máster	Especialista de CALISOFT	4 años

Tabla 9. Caracterización de los expertos.

Anexo 4: Encuesta realizada a los expertos para validar el procedimiento.

Cuestionario para validar la propuesta de procedimiento

1. ¿Considera Ud. Importante la aplicación de un procedimiento para definir requisitos de calidad en el proceso de desarrollo de software en la universidad?

   ____ Sí _____No

   ¿Por qué?

   _____

   _____

2. ¿Considera Ud. que la aplicación del procedimiento propuesto puede mejorar la definición de requisitos de calidad?

   ____ Sí _____No

   ¿Por qué?

   _____

3. Dé una valoración del 1 al 5 de los criterios expuestos a continuación según la propuesta a validar:

____ Necesidad del empleo de la propuesta.

____ Posibilidad de aplicación.

____ Aporte en el proceso de definición de requisitos de calidad en un proyecto.

____ Impacto en la Calidad de los Requisitos.

4. Evalúe el procedimiento propuesto según los siguientes aspectos.

a. Las actividades propuestas son:

___ Innecesarias

___ Necesarias pero no suficientes para definir requisitos de calidad

___ Necesarias y suficientes para definir requisitos de calidad

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 definir requisitos de calidad

___ Necesarios y suficientes para definir requisitos de calidad

Otras consideraciones al respecto:

_____

_____

d. ¿Considera el documento Excel definido en la investigación una buena forma para identificar y priorizar los requisitos de calidad?:

___ Si ___ No.

e. La asignación de roles y responsabilidades por fases del procedimiento es:

___ Innecesaria

___ Necesaria pero no suficiente

___Necesaria y suficiente

Otras consideraciones al respecto:

_____

_____

f. ¿Cómo considera la elección de las características y subcaracterísticas de calidad hecha para el desarrollo de esta investigación?

___ Sin valor

___ Carente de requisitos

___ Bastante completa

Otras consideraciones al respecto:

_____

_____

5. ¿Qué factores cree usted que podría atentar contra la correcta aplicación de este procedimiento en la UCI y cuáles lo facilitarían?

_____

_____

6. Luego de conocer el procedimiento valore la utilidad que este podría tener

__ No sería útil __ Pudiera ser o no __ Sería útil

7. Haga un comentario o aporte sobre el manual que usted está evaluando. (El comentario es libre y debe reflejar algún elemento de interés que aporte elementos a la mejora del mismo):

_____

_____

Anexo 5: Encuesta realizada a líderes de proyecto para recoger los problemas que tuvo el software luego de ser terminado.

Entrevista a líderes de proyecto para identificar los problemas del software luego de haberse terminado

Nombre del encuestado: _____

Proyecto: _____ **Cargo dentro del proyecto:** _____

Facultad: _____ **Tipo de aplicación**: _____

Preguntas:

1. Marque con una X. Una vez que estuvo instalado el software este presentó algún problema con:

   _____ El hardware

   _____ Conexión con otros sistemas

   _____ Seguridad del sistema

   _____ Poco entendimiento del usuario final.

   _____ Error en los resultados o cálculos

   _____ Otros

En caso de existir otros problemas especificar cuáles.

_____

_____

2. ¿Qué requisitos no funcionales no se tuvieron en cuenta?

___ Funcionalidad ___ Portabilidad

___ Usabilidad ___ Eficiencia

___ Confiabilidad ___ Mantenibilidad

3. ¿Qué requisitos no funcionales afectaron en mayor medida al software final?

___ Operabilidad ___ Coexistencia

___ Exactitud ___ Rendimiento

___ Seguridad ___ Utilización de recursos

___ Estabilidad ___ Instabilidad

___ Tolerancia ante fallos

4. ¿Cómo clasificó el software el cliente?

_____ Excelente _____ Bueno ____ Regular _____ Malo

5. En su opinión cuáles fueron los problemas más graves que tuvo el software. _____

GLOSARIO DE TÉRMINOS

- **BOOTSTRAP:** Estándar Europeo para Evaluación y Mejoras de Procesos de Desarrollo de Software.
- **Calisoft:** Centro de Calidad para Soluciones Tecnológicas. Garantiza el crecimiento continuo de una producción de software con calidad en la Universidad.
- **CITMATEL** (Tecnología de la Información y Servicios Telemáticos): desarrollo de sistemas dirigidos a automatizar la gestión en todo tipo de empresas.
- **CMM** (Capability Maturity Model o Modelo de Capacidad y Madurez): modelo de evaluación de los procesos de una organización.
- **CMMI** (Capability Maturity Model Integration o Modelo de Madurez y Capacidad Integrado): modelo de mejora de procesos en el desarrollo y mantenimiento del software.
- **ICSW:** Industria Cubana del Software.
- **IEEE** (Institute of Electrical and Electronics Engineers o Instituto de Ingenieros Electricistas y Electrónicos): Asociación técnico profesional dedicada a la estandarización.
- **ISO** (International Organization for Standardization u Organización Internacional para la Estandarización): organismo encargado de promover el desarrollo de normas internacionales.
- **MINREX:** Ministerio de Relaciones Exteriores de Cuba.
- **(NC) ISO\IEC 9126** (Norma Cubana ISO\IEC 9126): adaptación de la norma ISO\IEC 9126 a los procesos del país.
- **RUP** (Rational Unified Process o Proceso Unificado de Desarrollo de Software): proceso estructurado para el desarrollo de aplicaciones software. Conjunto de metodologías adaptables al contexto y necesidades de cada organización.
- **SEI** (Software Engineering Institute o Instituto de Ingeniería de Software): centro de investigación y desarrollo patrocinado por el departamento de Defensa de los Estados Unidos de América. Surge con el objetivo de desarrollar modelos de evaluación y mejoras en el desarrollo del software.
- **SOFTEL:** Empresa que ofrece soluciones informáticas para el sistema de salud.