Universidad de las Ciencias Informáticas
Facultad 8

Procedimiento Propuesto para medir la Calidad en la Gestión de Requisitos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Bedsy Guevara Mojena

Tutora: Msc. Ing. Haydée Mª Cruz Torres

Ciudad de la Habana, Julio 2007
"Año 49 de la Revolución"

"*Cuando puedas medir lo que estás diciendo y expresarlo con números, ya conoces algo sobre ello; pero cuando no puedas medirlo, cuando no puedas expresarlo con números, su conocimiento es precario e insatisfactorio: puede ser el comienzo del conocimiento, pero en sus pensamientos, apenas estás avanzando hacia el escenario de la ciencia***"**

Lord Kelvin, 1900

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __29__ días del mes de ___junio___del año ___2007__.

Bedsy Guevara Mojena

_____

Msc. Ing. Haydée Mª Cruz Torres

_____

DATOS DE CONTACTO

Ing. Haydée Mª Cruz Torres, Msc.
hmcruz@uci.cu
Facultad 8
Departamento de Ciencias Informáticas
Teléfono: 837 - 2476

Breve currículum

- Graduada en Ingeniería Informática en el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE), en 2001.
- Imparte docencia en universidades desde 2001.
- Profesora Instructora desde 2003.
- Profesora de Gestión e Ingeniería de Software.
- Profesora de Inteligencia Artificial.
- Máster en Informática Aplicada, desde 2005.
- Ha desarrollado diferentes trabajos investigativos, alguno de ellos: Gestión de Requisitos en el desarrollo Offshore, Modelamiento de Aplicaciones Web, Medición de la Calidad.
- Ha participado en diferentes eventos Nacionales e Internacionales de Informática que se realizan en nuestro país, como “Informática 2005” e “Informática 2007”

A mi madre, por ser mi todo…

A mi padre, que desde el cielo me ilumina con su luz…

Agradecimientos

- A mi mamá, por su eterno amor, por ser mi mejor amiga, por estar conmigo en todo momento y ser mi principal fuente de inspiración. A ella le debo lo que soy hoy.
- A mi familia y en especial a mi hermana y mi sobrino, por su interés y preocupación.
- A mi tutora, por iniciarme en el mundo de la investigación, por su constancia, sus regaños y sobre todo por regalarme una linda amistad…muy importante, por enseñarme la Habana.
- A todas mis compañeras del apto que me hicieron más amena mi estancia en la universidad, especialmente a Yainelis, Kenia y Yodaysis por estar más cerca de mí en los buenos y a veces malos momentos que compartimos.
- A todos mis compañeros de la UCI y la UHo[1]., por su apoyo cuando lo necesité y sus críticas constructivas que me ayudaron a ser mejor cada día.
- A los que leyeron mi tesis porque con sus comentarios y observaciones me permitieron enriquecerla.
- A Roset por su incondicional colaboración con la traducción, por saber escoger el lugar y hacer del encuentro el más agradable.
- A todos los profesores que durante todos estos años contribuyeron con mi formación profesional.
- A Cary y David por su apoyo y ánimo constante, además por permitirme ser como de la familia.
- A Aurora por sus sabios consejos y acogerme como una hija.
- A Marianne y familia por su cariño y dejarme usar su computadora cuando me hizo falta.
- A los Manueles y Jesusito, por confiar en mí y ser los mejores amigos de todos los tiempos.
- A mi hermanita de Nicaro, Ana Esther, por acortar la distancia con sus correos de aliento y esperanzas, pero más que eso, por hacer perdurar la amistad a pesar de las dificultades.
- A Yanelis por confiar siempre en mí y ser una amiga muy especial.
- A mi prima de Jagüey, Iliana, por su acogido y permitirme conocer personas maravillosas.
- Al team de Jagüey y Santo Suárez: Alexei, Reynier (el flaco), Noel y Darío por sus momentos de felicidad y toda la alegría que me propiciaron cuando más estresada estaba.
- Especialmente, a todo y todos los que hicieron de la UCI una quimera alcanzable.

Muchas Gracias!!!

[1] Universidad de Holguín “Oscar Lucero Moya”

Resumen

En el presente trabajo se realiza un estudio sobre los principales problemas que afectan al proceso de Gestión de Requisitos (GR), además se investigó sobre modelos y procesos de medición de la calidad, así como los atributos o características que se deben enfocar para medir la calidad en este proceso. Se propone y define un procedimiento cuantitativo que se caracteriza por ser integral, robusto y flexible para la medición y evaluación de la calidad del proceso y su respectiva Especificación de Requisitos del Software (ERS). El procedimiento, en sus siglas, RPQ (*Requirements Process Quality - Calidad en el Proceso Gestión de Requisitos*) realiza un aporte práctico al proponer un enfoque sistemático y disciplinado, propiciando la medición, evaluación, comparación y análisis de calidad en el proceso objeto de estudio. Se analizan las fases y actividades que componen al procedimiento, describiendo los procesos, modelos, técnicas, criterios y posibles herramientas a aplicar en dichas actividades. Por último se detallan las métricas que se proponen utilizar para evaluar el proceso y el producto respectivamente. El trabajo constituye una guía de ayuda para los desarrolladores de software a realizar sus especificaciones con la calidad que se requiere.

Palabras Clave: *Gestión de Requisitos*, *Especificación de Requisitos del Software, Calidad, Métricas.*

Índice

Índice de Figuras

Índice de Tablas

Introducción

Las instituciones desarrolladoras de software (SW) necesitan alcanzar una evolución y perfeccionamiento hacia una cultura de excelencia en la producción del software. Hoy en día las organizaciones deben ser capaces de desarrollar y entregar un software confiable, a tiempo y apegado al presupuesto acordado con el cliente, y los clientes quieren conocer con certeza que todo se consumará.

La mayoría de los ingenieros suelen tener la impresión de que los resultados que obtienen al resolver cualquier situación problémica son correctos, sin embargo, a pesar de que el sentido común junto con la experiencia personal pueden ser las armas más poderosas a su disposición, no deja de ser una percepción. El sentido común no es suficiente cuando se quiere comparar cuán buena es una solución frente a otra posible, o cuánto va a costar llevar a cabo cada una de las disyuntivas disponibles.

Por una parte, uno de los elementos fundamentales en el proceso de mejora del software, junto con la prevención de defectos, es su continua medición y de manera específica cómo se pueden utilizar dichas mediciones para tomar decisiones durante la evolución de un proyecto, así como plantear nuevos objetivos para mejorar su calidad [Orantes & Botello, 2006].

En el caso de las mediciones, para evaluar la calidad del software a través de los productos intermedios obtenidos en el ciclo de vida, el trabajo debe centrarse en analizar qué características [Orantes & Botello, 2006] medir para garantizar la calidad del producto final desde las etapas tempranas del ciclo de vida.

Independientemente de la norma o estándar que se adopte para regir la calidad del SW es necesario definir y realizar mediciones tanto para controlar como para evaluar el proceso de desarrollo de software con el objetivo de mejorar la calidad del mismo [Orantes & Botello, 2006].

Las medidas de los atributos internos del producto proporcionan al analista una visión general en tiempo real de la eficacia del análisis, del diseño, de la estructura del código, de la efectividad de los casos de prueba y de la calidad integral del software.

Es esencial que los ingenieros de software dispongan de mecanismos cuantitativos para estimar la calidad de los diseños y la efectividad de los programas [Orantes & Botello, 2006].

Uno de los aspectos fundamentales que influyen en la obtención de un software con calidad es la comunicación efectiva entre los usuarios, clientes y el equipo de desarrollo con el objetivo de llegar a un entendimiento de lo que hay que hacer. Durante muchos años e incluso en la actualidad, a pesar de las metodologías, normas, estándares, lineamiento, técnicas y herramientas que existen para ayudar a lograr un mejor resultado de las aplicaciones, muchos sistemas de software fallan (no se culminaron, se entregaron tarde, cuestan más de lo previsto, el producto resultante es difícil de usar o no se usaron) [Palomino, 2006] porque existieron incoherencias entre lo que el usuario quería, lo que realmente necesitaba, lo que interpretaba cada miembro del equipo de proyecto y lo que realmente se obtiene.

Por un lado, los requisitos representan las necesidades de los usuarios que deben ser satisfechas. En esencia, ellos son, el planteamiento del problema. Y en este aspecto es de vital importancia definir bien los objetivos del modelamiento del negocio.

Es necesario definir el término “requisito” que será empleado indistintamente en el presente documento. Este término ha sido concebido de varias maneras por diferentes autores aunque en sí, su significado es el mismo. Solo se hace referencia al concepto que proporciona la IEEE (*Institute of Electrical and Electronics Engineers - Instituto de Ingenieros Eléctricos y Electrónicos*), pues se considera que es una de las definiciones más completa.

De acuerdo con la IEEE un requisito es (a) una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo, (b) una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal, (c) una representación en forma de documento de una condición o capacidad como las expresadas en (a) o en (b) [IEEEComputer, 2004].

La forma más segura de garantizar que el software cumpla con lo establecido, es especificar detalladamente lo que se debe hacer y cómo se debe hacer, de esta manera se propicia un mecanismo

para respaldar el trabajo realizado ante cambios y reclamos en etapas posteriores. Además, la especificación de los requisitos le brinda al desarrollador y al cliente los medios de evaluar la calidad una vez que el software ha sido entregado [Cruz, 2005].

Por otra parte para lograr un mejor entendimiento y llevarse una idea clara de la situación actual de la crisis del SW existen valores cualitativos y cuantitativos que reflejan las principales causas de que los sistemas fallen o sean exitosos, los cuales se presentan a continuación. Estos valores fundamentan la importancia que en los últimos años se le ha dado a la Gestión de Requisitos (GR) como parte del proceso de desarrollo del software y la gran necesidad de medir la calidad con que fue ejecutado.

Existen algunos problemas con los requisitos, que fueron debatidos en la jornada[1] "**SOLO REQUISITOS 2006**" que se celebró en Madrid durante la semana del 23 al 27 de octubre del pasado año, como un punto de encuentro entre profesionales de la Ingeniería del Software, que incluyó 3 días de formación específica en temas relacionados con la Gestión e Ingeniería de Requisitos, entre otros temas de interés.

Las ponencias mostraron problemas como [Palomino, 2006]:

- Los requisitos NO reflejan las necesidades reales del cliente,
- son inconsistentes y/o incompletos (son miles),
- es caro cambiarlos una vez que se han pactado,
- se interpretan distintamente entre clientes, desarrolladores, gestores, personal de mantenimiento, etc.

Además reflejaron [Palomino, 2006] los siguientes datos estadísticos:

- Costo de la fase de requisitos: 10% - 15% del total proyecto y corregir un requisito es hasta 100 veces más caro que corregir un error de programación.
- El 31% de los proyectos software se cancelan antes de terminar.
- En grandes compañías, solamente el 9% de los proyectos se entregan en tiempo y forma.
- Fallar en los requisitos, significa fallar en todas las fases del ciclo de vida.

[1] http://www.calidaddelsoftware.com/modules.php?name=News&file=article&sid=187

Es importante realizar un análisis comparativo teniendo en cuentas las principales causas de fracaso y éxito de los sistemas software en el año 1997 [InformeQSS, 1997] y en el año 2004 [InformeSEI, 2004] [Palomino, 2006].

Las principales causas de que los sistemas fallen en el año 1997 [InformeQSS, 1997] se representan en la Figura 1.

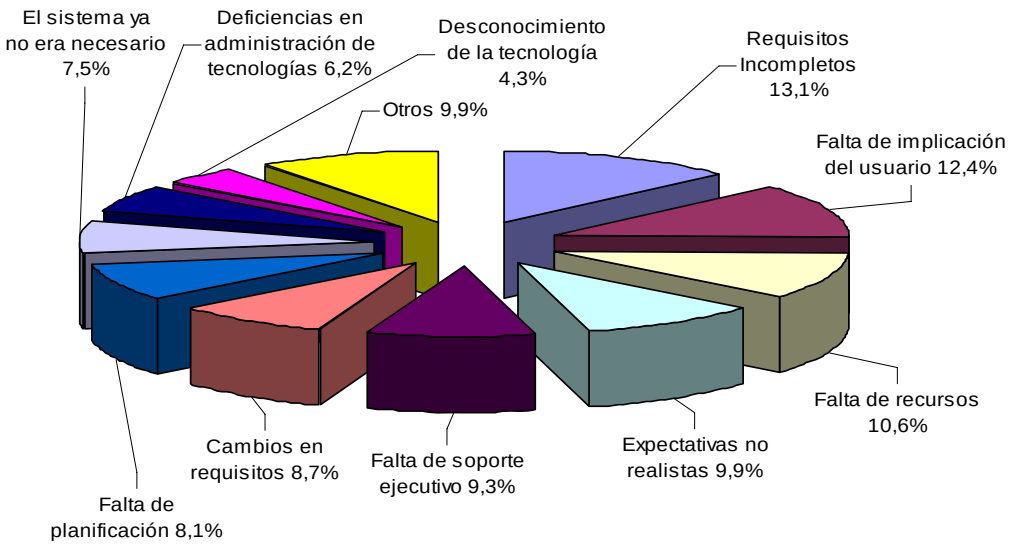

Figura 1 Factores claves en el fallo de la ejecución de proyectos (1997)

Si se analizan los factores claves de fallo representados anteriormente se puede percibir claramente que el mayor peso que conlleva al fracaso de los proyectos de software lo constituyen precisamente los requisitos inconclusos, seguido de la falta de intervención de los usuarios, por lo tanto se puede concluir que estos dos factores principalmente son aspectos muy importantes a tener en cuenta durante el proceso de desarrollo del SW que tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente.

Mientras que los factores claves de éxito de los proyectos software en el año 1997 [InformeQSS, 1997] se representan el la Figura 2.

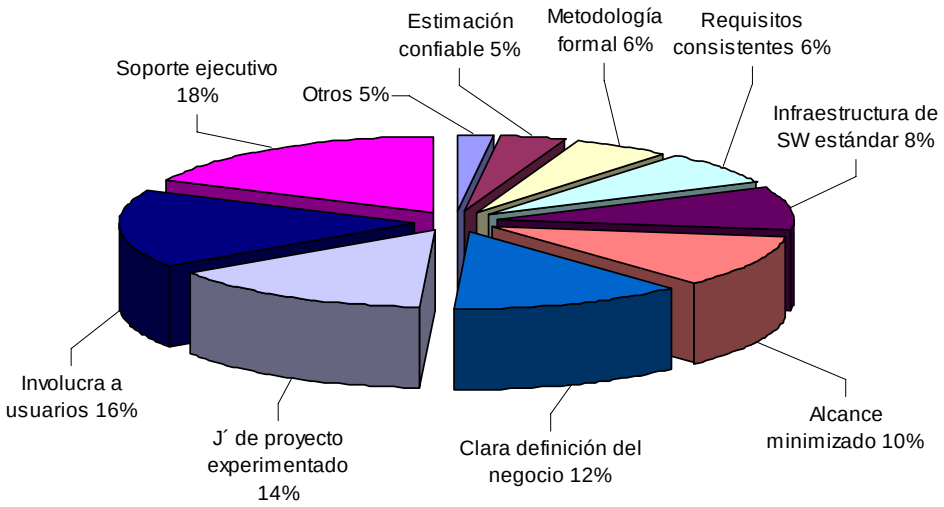

Figura 2 Factores claves en el éxito de la ejecución de proyectos (1997)

Se puede observar claramente (Figura 2) que el factor que involucra a los usuarios se repite, con uno de los mayores pesos, esto quiere decir que todo cuanto se haga debe estar orientado a lograr una especificación de requisitos con calidad e incluir en todo momento a los usuarios para poder ofrecerles lo que en realidad ellos necesitan y esperan.

Por otra parte, si se analiza la siguiente información estadística más reciente [InformeSEI, 2004] [Palomino, 2006]:

- Los problemas con los requisitos son la PRIMERA causa de proyectos cancelados o fuera de presupuesto.
- 42-64% de todos los defectos se originan por los requisitos.
- 25-40% del total de los gastos del proyecto se atribuyen a volver a realizar trabajos por culpa de los requisitos defectuosos.
- Reparar los defectos durante la fase de construcción cuesta de 5-20 veces más que preverlos durante el análisis de los requisitos.

Entonces se puede concluir que a pesar del transcurso de los años no se ha avanzado cualitativa y cuantitativamente en función de lograr obtener una buena Gestión de Requisitos y por ende una

especificación con la calidad que se requiere para lograr sistemas automatizados mejores. Los requisitos siguen siendo el factor común de las principales causas del fracaso.

Actualmente se conocen métodos, herramientas y procedimientos para llevar a cabo la GR, pero debido a su complejidad este proceso continúa afectando el exitoso desarrollo de los productos software. Existe una tendencia cada vez mayor de controlar este proceso, sin embargo no se aprovechan la infinidad de datos que se pueden recopilar en función de lograr detectar las causas que provocan las deficiencias que se presentan.

Para minimizar la complejidad y relatividad inherentes al concepto de calidad en el proceso de GR, se hace necesaria la aplicación de modelos que permitan estimar la calidad del mismo a través de métricas que admitan recopilar datos de forma general.

Un análisis exhaustivo de estos datos puede ayudar a mejorar considerablemente la calidad de este proceso, pues se pueden tomar medidas con el objetivo de desarrollar determinadas habilidades en los analistas. No existe un procedimiento que oriente dónde y cómo registrar esta información de una manera provechosa y eficiente. Además, los responsables de estos procesos no están conscientes de las ventajas que les puede proporcionar contar con un método que refleje la calidad de su trabajo.

Dada la **Situación problémica** planteada anteriormente el presente trabajo surge por la necesidad de darle la mejor solución posible, por lo que el **Problema Científico** radica en ¿cómo analizar los datos recopilados en la Gestión de Requisitos de manera que permita evaluar la calidad de este proceso?

El **Objeto de esta Investigación** son las Métricas de Calidad y el **Campo de Acción** definido en el que se enfoca la investigación son las Métricas para evaluar la calidad en la Gestión de Requisitos.

Para proporcionarle una solución al problema analizado se define como **Objetivo General:** Proponer un procedimiento que ayude a los responsables del proceso de Gestión de Requisitos a evaluar constantemente la calidad del proceso y detectar sus deficiencias.

Del objetivo global se derivaron los siguientes **Objetivos específicos** que ayudan a darle un eficiente cumplimiento:

- Definir qué datos recopilar durante el proceso de revisión al proceso.
- Definir modelos de calidad, procesos y técnicas de medición para el procedimiento.
- Determinar qué métricas se pueden emplear para evaluar la calidad del proceso y del producto.

Partiendo de la siguiente **Hipótesis:**
Proponiendo un método para registrar y analizar datos durante la captura de requisitos se puede lograr un mayor control de las deficiencias permitiendo tomar decisiones para mejorar este proceso.

A continuación se enmarcan un conjunto de **Tareas de investigación** que se proponen solucionar cada uno de los objetivos específicos determinados y de alguna que otra forma probar la hipótesis definida:

- Investigación sobre el proceso de Gestión de Requisitos.
- Investigación sobre procesos de revisión.
- Investigación sobre procesos de medición para definir un método que indique cómo registrar y analizar los datos recopilados.
- Investigación sobre Modelos y Métricas de Calidad.
- Análisis de las métricas que pueden utilizarse en la Gestión de Requisitos.
- Diseño de plantillas donde se registren los datos.
- Enumeración de las posibles deficiencias que pueden aparecer en este proceso.
- Recomendación de soluciones a estas deficiencias.

Entre los **Métodos Científicos** que mayormente guían el desarrollo de esta investigación y de acuerdo a su forma de cognición se emplearon combinaciones de los *métodos teóricos* y *métodos empíricos*. Los **Métodos teóricos** utilizados fueron los siguientes:

- ***Analítico – sintético*.** Al buscar la esencia del problema de investigación y los rasgos que lo caracteriza y distingue; extrayendo los elementos más importantes que se relacionan con el objeto de investigación, descomponiéndolos por separados y profundizando en el estudio de cada uno de ellos para sintetizarlos en la solución de la propuesta.

- ***Inductivo – deductivo.*** A la hora de elaborar la hipótesis central de la investigación y proponer un procedimiento nuevo a partir de los resultados parciales.
- ***Histórico – lógico.*** Al hacer un estudio crítico del los trabajos anteriores en este contexto y para utilizarlos como punto de referencia y comparación de los resultados alcanzados, además para constatar teóricamente cómo ha evolucionado la investigación en un período de tiempo, en toda su trayectoria o en un fragmento temporal de la lógica de su desarrollo.
- ***Modelación.*** Al descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio, explicando las razones por las cuales el procedimiento propuesto es el que más se ajusta a las características del proceso de Captura de Requisitos.

Conjuntamente a los métodos mencionados anteriormente y para completar la investigación se hizo necesario hacer uso de los siguientes **Métodos empíricos**:

- **Observación.** Al recoger la información de cada uno de los conceptos o variables definidas en la hipótesis, además para investigar el problema de investigación en su manifestación externa, por lo que es importante utilizar métodos como son la medición y la experimentación.
- **Experimento.** Para obtener conocimiento sobre el objeto de estudio, verificar la hipótesis y el procedimiento propuesto.
- **Medición.** Al obtener información numérica a partir de los atributos de calidad y analizar los datos numéricos que se obtienen, trazando una estrategia que permita la toma de decisiones en función de lograr una mejora en la Captura de Requisitos como proceso.

Este trabajo se encuentra estructurado en tres capítulos con la intención de realizar una división por los contenidos que serán tratados. En el primer 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. En el segundo capítulo se presenta el procedimiento propuesto compuesto por cuatro fases que contienen un conjunto de métodos, proceso, modelos y técnicas para medir y evaluar la calidad del proceso de Gestión de Requisitos y la Especificación de Requisitos del Software como producto. Por último, en el tercer capítulo se presentan y se analizan los resultados obtenidos después de aplicada la fase de Medición al proceso del procedimiento propuesto a un proyecto real, estudiando las principales causas y posibles medidas a tener en cuenta.

Capítulo 1. Fundamentación Teórica

1.1 Introducción

Después de haberse hecho un análisis general de la situación actual en el desarrollo del software y haber introducido los aspectos más problemáticos que entorpecen su calidad, este capítulo servirá para fundamentar el estado del arte a través de la revisión bibliográfica realizada. Se abarcan los conceptos fundamentales a tener en cuenta, empezando por el ambiente de la calidad, como principal objetivo a la hora de desarrollar un producto software; la Gestión de Requisitos de forma general, analizando cuidadosamente los problemas que se pueden encontrar en este proceso además del proceso de Captura de Requisitos; se analiza el ámbito de las métricas, haciendo énfasis en las métricas de calidad que posibiliten evaluar el proceso antes mencionado y su respectiva especificación.

1.2 Gestión, Control y Aseguramiento de la Calidad del Software

Lamentablemente algunos desarrolladores de software todavía siguen pensando que la calidad de software es algo a tener en cuenta y de preocuparse cuando ya han generado el código, nada menos cierto y lejos de la realidad [Pressman, 2002]. A continuación se hace necesario presentar algunos conceptos con respecto a la calidad para fundamentar y lograr entender temas más específicos que serán tratados más adelante en este apartado.

1.2.1 Calidad

El diccionario *American Heritage Dictionary*, define la calidad como una “característica o atributo de algo”, es decir, que como un atributo de un elemento, la calidad se refiere a las características mensurables que se pueden comparar con estándares como la longitud, color, etc. [Pressman, 2002]. Sin embargo, el software en su gran extensión, como entidad intelectual, es mucho más difícil de caracterizar, esto no quiere decir que no se pueda hacer, sino que es mucho más complejo definir las medidas de características de un software [Pressman, 2002]. Cuando se tiene en cuenta estas características mensurables, se pueden encontrar dos enfoques de calidad: calidad del diseño y calidad de concordancia. Esto quiere decir que en el desarrollo del software, la *calidad de diseño* comprende los requisitos, especificaciones y el diseño del sistema mientras que la *calidad de concordancia* es un aspecto centrado

principalmente en la implementación. Entonces si la implementación sigue el diseño y el sistema resultante cumple los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta [Pressman, 2002].

Por su parte, la ISO[1] (*International Organization for Standardization – Organización Internacional para la Estandarización*), agrega otros aspectos importantes a tener en cuenta cuando define la calidad como el conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas [ISO/TC 176/SC1N322, 2007].

1.2.2 Control de calidad

La [ISO/TC 176/SC1N322, 2007], afirma que el *Control de calidad* es un conjunto de técnicas y actividades de carácter operativo, utilizadas para verificar los requisitos relativos a la calidad del producto o servicio.

Por otra parte, Pressman aborda con más profundidad este tema, al exponer que el *Control de calidad* es una serie de inspecciones, revisiones y pruebas utilizadas en el proceso del software para asegurar que cada producto cumpla con los requisitos que le han sido asignados. Incluye un bucle de retroalimentación (feedback) del proceso que creó el producto. La combinación de medición y realimentación permite afinar el proceso cuando los productos de trabajo creados fallan al cumplir sus especificaciones [Pressman, 2002].

Al mismo tiempo afirma que un concepto clave del control de calidad es que se hayan definido todos los productos y las especificaciones mensurables en las que se puedan comparar los resultados de cada proceso. El bucle de realimentación es esencial para reducir los defectos producidos [Pressman, 2002].

[1] ISO no es un acrónimo; proviene del griego ***iso***, que significa igual. Es un error común el pensar que ISO significa International Standards Organization, o algo similar; en inglés su nombre es International Organization for Standardization, mientras que en francés se denomina Organisation Internationale de Normalisation; el uso del acrónimo conduciría a nombres distintos: IOS en inglés y OIN en francés, por lo que los fundadores de la organización eligieron ISO como la forma corta y universal de su nombre.

1.2.3 Aseguramiento de la Calidad del Software

El Aseguramiento de Calidad del Software, en inglés Software Quality Assurance (SQA), expresa la [ISO/TC 176/SC1N322, 2007], es un conjunto o patrón de acciones planificadas y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisfará los requisitos establecidos sobre calidad, Schoonmaker coincide en su totalidad con este concepto [Pressman, 2002]. Mientras que Donald J. Reifer y Pressman agregan que también existen actividades de protección [Pressman, 2002], estimación y supervisión de las actividades de desarrollo, que se realizan de forma independiente al equipo de desarrollo [de Antonio, 2002].

[Pressman, 2002] y el Dr. Edward H. Bersoff [de Antonio, 2002] la definen como un conjunto de procedimientos, técnicas y herramientas, aplicados por profesionales, durante el ciclo de desarrollo de un producto, para asegurar que el producto satisface o excede los estándares o niveles de calidad preestablecidos.

Por otro lado, Roger Pressman agrega que el Aseguramiento de la calidad consiste en auditorías y funciones de información de la gestión. El objetivo principal que tiene es proporcionar la gestión para informar de los datos necesarios sobre la calidad del producto, por lo que va adquiriendo una visión más profunda y segura de que la calidad del producto está cumpliendo sus objetivos [Pressman, 2002].

Otro aspecto importante que se debe considerar [Pressman, 2002] es que el SQA es responsabilidad en alguna medida de todos los que integran una organización y sobre todo este grupo que lleva a cargo el SQA debe representar al cliente, es decir, deben mirar el software desde el punto de vista del cliente.

1.2.3.1 Actividades de SQA

Como se había mencionado anteriormente, el Control de la Calidad se centra fundamentalmente en el producto, sin embargo el SQA también va a comprender el proceso de desarrollo. Estas dos grandes actividades van a estar estrechamente relacionadas, porque por un lado el grupo de Aseguramiento de Calidad es responsable de establecer el tipo de Control de Calidad que se va a realizar, aunque por lo general no es responsable de efectuar esos controles, de esto se encarga el equipo de desarrollo. Por otro

lado, el grupo de Aseguramiento de Calidad va a aprovechar los resultados del control de calidad para evaluar y mejorar el proceso de desarrollo, con el objetivo de conseguir productos de mayor calidad.

Por lo tanto, las actividades fundamentales que realiza o facilita el grupo de SQA son [de Antonio, 2002]:

- Planificación de la calidad: Consiste en seleccionar, clasificar y ponderar las propiedades de calidad que se van a establecer como requisitos, con respecto al producto y con respecto al proceso. Se elegirán también los mecanismos de control de calidad a utilizar para medir y evaluar estas características, además de determinar las metas a alcanzar.
- Supervisión de la calidad: Consiste en supervisar y corregir, si es necesario, el trabajo que se está realizando (según los resultados obtenidos con las actividades de control de calidad), con el objetivo de llegar a satisfacer los requisitos establecidos.
- Construcción de la calidad: Actividades constructivas son aquellas que sirven para "construir" la calidad, es decir, son actividades preventivas cuyo objetivo es evitar la introducción de errores mediante la puesta en práctica de ciertos principios, métodos, formalismos y herramientas.

Por otro lado, el Instituto de Ingeniería del Software (SEI) [Pressman, 2002] recomienda un grupo de actividades más amplio, que se enfrentan con la planificación de aseguramiento de calidad, supervisión, mantenimiento de registros, análisis e informes, estas son:

- Establecimiento de un plan de SQA para un proyecto.
- Participación en el desarrollo de la descripción del proceso de software del proyecto.
- Revisión de las actividades de ingeniería del software para verificar su ajuste al proceso de software definido.
- Auditorías de los productos de software designados para verificar el ajuste con los definidos por parte del proceso de software.
- Asegurar que las desviaciones del trabajo y los productos del software se documentan y se manejan de acuerdo con un procedimiento establecido.
- Registrar lo que no se ajuste a los requisitos e informar a sus superiores.
- Coordinar el control y la gestión de los cambios.
- Recopilar y analizar las métricas del software.

De forma general estas actividades se resumen en: un enfoque de gestión de la calidad; tecnología de ingeniería de software efectiva (métodos y herramientas); revisiones técnicas formales (RTF) que se aplican durante el proceso del software; una estrategia de prueba multiescalada; el control de la documentación del software y de los cambios realizados; un procedimiento que se ajuste a los estándares de desarrollo del software cuando sea posible y por último un mecanismo de medición y de generación de informes [Pressman, 2002].

A continuación se hace una breve descripción de las revisiones.

1.2.3.2 Revisiones

Se puede definir una Revisión como una reunión formal en la que se presenta el estado actual de los resultados de un proyecto a un usuario, cliente u otro tipo de persona interesada, y se realiza un análisis estructurado de los mismos [de Antonio, 2002]. También se les considera un “filtro” para el proceso de Ingeniería de Software, se aplican en varios momentos del desarrollo de un software y sirven para detectar defectos que puedan ser eliminados, también para “purificar” actividades como el análisis, diseño y codificación.

Las revisiones son, hoy en día, el único método de control de calidad eficaz [de Antonio, 2002] en las fases iniciales del desarrollo a la hora de identificar desviaciones con respecto a las especificaciones de calidad. Asimismo las revisiones redundan en una mejora directa de la calidad del objeto que se examina y provocan, indirectamente, una mejora de la calidad del proceso de desarrollo, al facilitar la comunicación entre los miembros del equipo de desarrollo. Al mismo tiempo facilitan el control del costo y tiempo.

Dentro de las revisiones se encuentran las Inspecciones, que es donde los participantes van leyendo el documento, paso a paso, guiados por el autor del mismo, y comprobando en cada paso el cumplimiento de los criterios de una Lista de chequeo (Checklist) o Listas de comprobación como también se les conocen y constituyen técnicas que se utilizan para realizar las inspecciones. Seguidamente se describe una lista de chequeo.

1.2.3.3 Lista de Chequeo

Es una técnica que requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en la propia Lista de chequeo. Esta lista será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista [Escalona & Koch, 2002]. Al intentar dar respuesta a las preguntas formuladas deben salir a la luz los problemas que puedan existir. Cada revisor debe completar la lista y anotar cualquier tipo de pregunta o defecto detectado [de Antonio, 2002].

1.2.4 Gestión de calidad

Así mismo, la [ISO/TC 176/SC1N322, 2007] explica que la Gestión de la calidad trata aspectos acerca de la función de gestión que determina, aplica la política de la calidad, así como objetivos y responsabilidades y que lo realiza con medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento de calidad y la mejora de la calidad.

La Gestión de la calidad es responsabilidad de todos los niveles ejecutivos, pero debe estar guiada por la alta dirección. De igual manera afronta los problemas y aplica los recursos necesarios para resolver problemas identificados por el Aseguramiento de la calidad. Su realización involucra a todos los miembros de la organización. En la Gestión de la calidad, se tienen en cuenta también criterios de rentabilidad [Pressman, 2002].

1.3 Calidad del software

El interés por la calidad del software crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades.

Si se desea mejorar en cuanto a la calidad del software entonces es preciso definir la calidad y sobre todo medirla. Todavía el mayor problema en la Gestión de la Calidad es que el mismo término “calidad” es ambiguo, tanto que comúnmente es mal entendido. La confusión puede ser atribuida a muchas razones. Primero, la calidad no es una única idea, pero sí un concepto multidimensional. Las dimensiones de la

calidad incluyen la entidad de interés, el punto de vista de esa entidad y los atributos de calidad de esa entidad. Segundo, para cualquier concepto hay niveles de abstracción; cuando las personas hablan acerca de calidad, una parte puede estar refiriéndose en su sentido más amplio, mientras que otra parte puede referirse a un significado más específico. Tercero, el término calidad es una parte del lenguaje diario, su uso es popular y profesional al mismo tiempo, que puede ser muy diferente para cada caso [Kan, 2002].

A la hora de delimitar la calidad del software se pueden adoptar diferentes aproximaciones. Como primera aproximación es importante diferenciar entre la calidad del producto software y la calidad del proceso de desarrollo de éste (calidad de diseño y fabricación) [de Antonio, 2002].

No obstante, las metas que se establezcan para la calidad del producto van a determinar los objetivos a establecer para la calidad del proceso de desarrollo, ya que la calidad del primero va a depender, entre otros aspectos, del segundo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

También es fundamental recalcar que la calidad de un producto software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, códigos, y otros). No basta con verificar la calidad del producto una vez finalizado cuando los problemas de mala calidad ya no tienen solución o su reparación es muy costosa [de Antonio, 2002].

Los desarrolladores de SW se encuentran con diferentes problemas a la hora de hablar de la calidad de un producto software [de Antonio, 2002], dentro de estos se encuentran: (1) la definición misma de la calidad del software [Kan, 2002], ya que se hace difícil encontrar un conjunto de propiedades que den una indicación de su calidad y para dar solución a este problema surgieron los *modelos de calidad*; (2) la comprobación de la calidad del software, pues muchas veces no se sabe medir el grado de calidad de un producto software y para ello juega un papel fundamental el *control de calidad*; (3) la mejora de la calidad del software, en este sentido se hace dificultoso saber utilizar la información disponible acerca de la calidad del producto software para mejorar su calidad a lo largo del ciclo de vida, en este eje aparecen dos conceptos importantes, ellos son la *gestión de calidad* y el *aseguramiento de la calidad* [de Antonio, 2002].

A continuación se exponen diferentes definiciones de *Calidad del Software:*

La *Calidad del software* es el grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario [IEEE 610, 1990].

Por su parte, la [IEEE 729, 1983] expresa que es el grado con el cual el cliente o usuario percibe que el software satisface sus expectativas.

Según Pressman, la *Calidad del software* se define como 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, 2002].

La autora de este trabajo comparte el mismo criterio de Roger Pressman al afirmar que esta definición sirve para hacer hincapié en tres puntos importantes [Pressman, 2002]:

1. Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad[1].
2. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
3. Existe un conjunto de requisitos implícitos que a menudo no se mencionan, por ejemplo: el deseo por facilitar el uso y un buen mantenimiento. Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software queda en entredicho.

Una conclusión evidente a la que se puede llegar a partir de estas definiciones es que la calidad es algo relativo. Siempre va a depender de los requisitos o necesidades que se desee satisfacer. Por eso, la

[1] Es importante resaltar que la calidad se extiende a los atributos técnicos de los modelos de análisis, de diseño, y codificación. Los modelos que presentan una alta calidad darán lugar a un SW de una alta calidad desde el punto de vista del cliente.

evaluación de la calidad de un producto siempre va a implicar una comparación entre unos requisitos preestablecidos y el producto realmente desarrollado.

Teniendo esto en cuenta, en un producto software se van a tener diferentes visiones de la calidad: necesaria o requerida (la que quiere el cliente); programada o especificada (la que se ha especificado explícitamente y se intenta conseguir) por último la realizada (la que se ha conseguido).

A la intersección entre la *calidad requerida* y la *calidad realizada* se le llama *calidad percibida*, y es la única que el cliente valora. Toda aquella calidad que se realiza pero no se necesita es un gasto inútil de tiempo y dinero. En esto concuerda Robert Glass al afirmar que la calidad es importante pero si el usuario no queda satisfecho, ninguna otra cosa realmente importa [Pressman, 2002] y en este sentido es de vital importancia lograr una entrega dentro de presupuesto y del tiempo establecido para alcanzar un producto software íntegro y de alta calidad.

No basta con las definiciones antes mencionadas que pueden resultar un tanto generales e imprecisas a la hora de construir un software de alta calidad, por lo que se hace necesario introducir el tema de los Modelos de Calidad, que ayudan a definir la calidad del software de una forma más precisa y provechosa.

1.4 Modelos de calidad del software

Diferentes literaturas coinciden en que un modelo de calidad 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] [Buglione & Abran, 1999].

Unos de los modelos de calidad más antiguos y extendidos es el de McCall [de Antonio, 2002], y de él han derivado otros modelos, como el de Boehm y otros más hasta llegar al propuesto por el estándar ISO/IEC 9126 que será abordado en un próximo epígrafe.

Implantar modelos de calidad tiene como objetivo principal que las instituciones desarrollen sistemáticamente, productos, bienes y servicios de mejor calidad y cumplan con las necesidades y deseos de los clientes.

Algunos de los modelos de calidad del software encontrados durante esta investigación se mencionan a continuación [de Antonio, 2002]:

- ♠ Modelos de Calidad para el Proceso
  - o Modelo *CMM* [1] - Capability Maturity Model (1993) y *CMMI* [2]- Capability Maturity Model Integration (2001). Desarrollados por el SEI (Software Engineering Institute - Instituto de Ingeniería del Software). Son modelos con mayor repercusión internacional.
  - o *Tickit* [3] (BSI - ISO 9001:2000 para desarrollo de SW) como referencia fundamentalmente en el Reino Unido.
  - o *ISO/IEC 90003:2004* [4] como guía de apoyo a organizaciones adoptando el modelo ISO 9001:2000 para desarrollo de software.
  - o *EFQM* [5] - European Foundation for Quality Management como modelo para empresas en busca de la excelencia empresarial.
  - o *ISO/IEC 15504* [6] (SPICE - Software Process Improvement and Capability dEtermination) como marco general de convergencia para todos los modelos y métodos de evaluación de procesos.
  - o *Paradigma GQM* [7] - Goal-Question-Metric [Basili y Rombach, 1988]
- ♠ Modelos de Calidad para el Producto
  - o *Modelo FCM* [8]- Factors-Criteria-Metrics [McCall, 1977]. Es uno de los más antiguos y extendidos en la actualidad.
  - o *Modelo de Boehm* [9] [Boehm, 1978]. Se deriva del anterior.
  - o *Marco ISSO/IEC 9126* [10] [ISSO/IEC, 1991]. Define 6 características deseables en cualquier producto software.

---

[1] http://www.sei.cmu.edu/cmm/
[2] http://www.sei.cmu.edu/cmmi/
[3] http://www.tickit.org/
[4] http://www.praxiom.com/iso-90003.htm
[5] http://www.efqm.org/
[6] http://dmi.uib.es/~bbuades/calidad/sld037.htm
[7] http://www.sel.unsl.edu.ar/ApuntesMaes/2004/CursoOlsina/Parte2.2-Metricas-Heuris.pdf
[8] http://www.sc.ehu.es/jiwdocoj/remis/docs/mmg-2000.ppt & http://www.monografias.com/trabajos5/call/call.shtml
[9] http://es.wikipedia.org/wiki/Espiral_de_Boehm
[10] http://www.iso.org/ & http://en.wikipedia.org/wiki/ISO_9126

- *ISO/IEC 14598* [1], que define procesos de evaluación de productos (componentes o sistemas completos) desde diversas perspectivas (desarrollador, comprador, evaluador), en combinación con ISO/IEC 9126.

1.4.1 Estructura de los modelos de calidad

La estructura de un modelo de calidad se puede representar de forma jerárquica o arbórea. Es un concepto que se deriva de un conjunto de sub-conceptos, los cuales se van a evaluar a través de un grupo de métricas o indicadores en diferentes etapas y que ambos van a ser aplicables para predecir-asegurar-verificar el cumplimiento de las metas definidas antes-durante-después de la producción del producto software [Buglione & Abran, 1999], en esto radica su mayor beneficio. Por lo general tienen una estructura en tres niveles como se representa en la Figura 1.1 [de Antonio, 2002], aunque se pueden distinguir entre ellos por la cantidad de niveles (Ver Tabla 1.1) [Buglione & Abran, 1999].

En esta investigación se va a utilizar la terminología ISO porque el modelo que se presenta fue escogido de esta organización. En el nivel más alto de la jerarquía se encuentran las características de calidad (CC) definidas a partir de la visión del usuario del software, y conocidos también como atributos de calidad externos, como se muestra en la Figura 1.1. Cada una de las CC se descompone en un conjunto de sub-características (SC), o sea aquellos atributos que cuando están presentes contribuyen a obtener un software de calidad. Se trata de una visión de la calidad técnica, desde el punto de vista del producto software y se les denomina también atributos de calidad internos. Finalmente para cada una de las SC se definen un conjunto de métricas (MC) o medidas cuantitativas de ciertas características del producto que indican el grado en que dicho producto posee un determinado atributo de calidad.

[1] http://www.albertolacalle.com/hci_estandares.htm

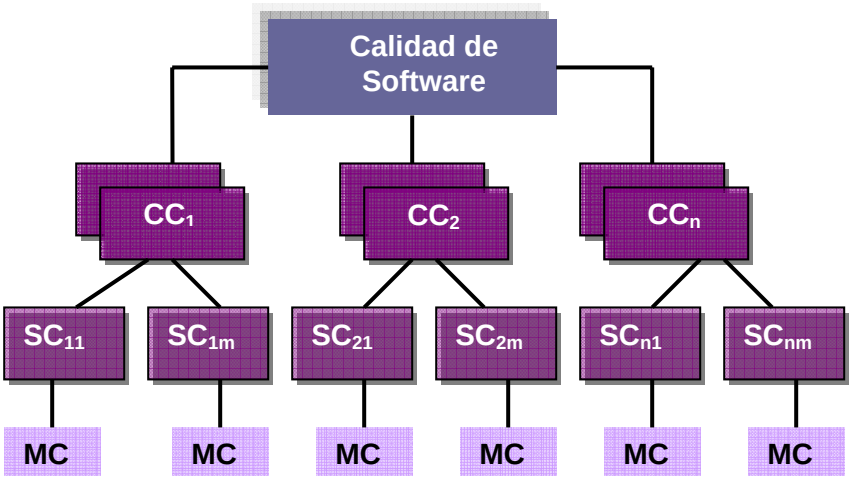

Figura 1. 1 Estructura de los Modelos de Calidad

De esta manera, se concretan los aspectos relacionados con la calidad a través de un modelo de forma tal que se puede definir, medir y planificar. Además el empleo de un modelo de calidad permite comprender las relaciones que existen entre diferentes características de un producto software [de Antonio, 2002].

Para clarificar la definición de los modelos de calidad se pueden considerar dos puntos de vistas diferentes. Teniendo en cuenta el *número de niveles*, se pueden encontrar modelos con dos o tres niveles, tal como se muestra en la Tabla 1.1 y considerando el *tipo de relación* que existe entre cada uno de los niveles, pueden ser una relación de **1: n** (uno a muchos*)* en el caso del marco ISO/IEC 9126, donde sus características de alto nivel se descomponen en un conjunto de Sub-características de bajo nivel, mientras que el otro tipo de relación puede ser de **n: m** (muchos a muchos) en el caso del modelo FCM (Factor-Criteria-Metric) de McCall donde cada sub-característica se relaciona con más de una característica [Buglione & Abran, 1999].

Puede pensarse que esta definición de modelo de calidad es comparable con el paradigma GQM (Goals-Questions-Metrics – Metas-Preguntas-Métricas) de Basili y Rombach que también será tratado más adelante en este apartado, donde las Metas y Preguntas (los dos primeros niveles) pueden ser

equivalente a las Características y Sub-características mientras que las métricas están referenciadas en un contexto más técnico [Buglione & Abran, 1999].

La Tabla 1.1 muestra las diferentes terminologías empleadas por los distintos autores para definir sus modelos de calidad.

Tabla 1. 1 Modelos de Calidad del Software

Nivel	***McCall***	***Boehm***	***ISO***	***IEEE***	***Dromey***
1	Caract. Alto Nivel	Factor	Característica	Factor	Atributo de Alto Nivel
2	Caract. primitivas	Criterio	Sub-característica	Sub-factor	Atributo subordinado
3[16]	(Métrica)	(Métrica)	(Métrica)	Métrica	

Por otro lado, en contra de los modelos de calidad pesa que aún no ha quedado demostrada la validez absoluta de ninguno de ellos [de Antonio, 2002]. Además las conexiones que se establecen entre características, atributos y métricas se derivan de la experiencia, y de ahí que existan múltiples modelos, aunque sería interesante que cada organización adecuase su modelo de calidad a sus necesidades y perspectivas [de Antonio, 2002].

1.4.2 Paradigma GQM (Goal - Question - Metric)

Antes de entrar a analizar el Paradigma GQM (Goal-Question-Metric – Meta-Pregunta-Métrica) de Basili y Rombach creado en 1988 [Palacio, 2006] [de Antonio, 2002], sería prudente destacar que los factores que afectan la calidad del software se pueden categorizar en dos amplios grupos: (1) factores que se pueden medir directamente (por ejemplo: defectos por puntos de función o elementos de especificación) y (2) factores que se pueden medir solo indirectamente (por ejemplo: facilidad de uso o de mantenimiento). En los dos casos debe estar presente la medición. Se debe comparar el software (documentos, programas, datos) con una referencia y llegar a una conclusión sobre la calidad [Pressman, 2002].

De acuerdo a investigaciones realizadas [Olsina, 1999], para que las métricas sean efectivas deben estar focalizadas en metas específicas, aplicadas a todo o parte del ciclo de vida de los entes (procesos,

[16] El paréntesis para “Métrica” en la tercera fila de la tabla indica que la arquitectura del modelo no menciona formalmente este nivel, incluso si existe y se necesita para hacer evaluaciones.

productos y/o artefactos, recursos) e interpretadas en función de la comprensión del contexto organizacional. Esto implica que las mediciones se deben definir bajo una estrategia de arriba hacia abajo, o también conocida como “top-down”. A partir de estas premisas surge el modelo Goal-Question-Metrics (GQM) de Basili y se describe en este apartado.

A pesar de que este modelo fue creado hace muchos años tiene mucha vigencia, brinda una facilidad de compresión que lo hacer ser un modelo muy práctico, por lo tanto se puede aplicar sin muchos problemas. Además ofrece una flexibilidad que posibilita adecuarlo a las características específicas de cada producto software. Se basa en la mejora de la definición clara de procesos y productos. Propone una estructura que permite alcanzar los objetivos cruciales del proyecto, obtener las métricas respectivas con el análisis e interpretación de los datos y plantea un enfoque de medición para evaluar la calidad del software basado en la identificación de objetivos a lograr, que es una de las formas más eficientes de definir qué medir y cómo medirlo [Palacio, 2006] [IEEE 1061, 1998] [Olsina, 1999].

Este modelo, como muestra la Figura 1.2 presenta un proceso de medición orientado a los objetivos, donde cada métrica está dirigida a medir una meta u objetivo. La idea es que se debe tener buenas razones para recoger datos. Cada objetivo debe contestarse con una o varias preguntas. La pregunta se establece de forma que la métrica pueda responderla claramente. Por último, la métrica o indicador es una entidad cuantitativa que dé respuesta a una pregunta especifico [IEEE 1061, 1998].

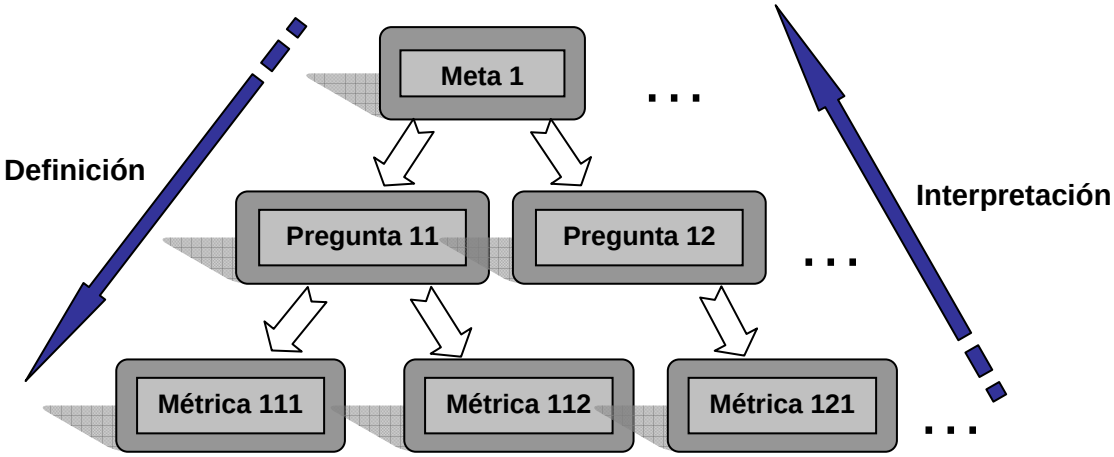

Figura 1. 2 Paradigma GQM de Basili & Rombach (1988)

De esta forma, los tres niveles que lo conforman son los siguientes [Laird & Brennan, 2006] [IEEE 1061, 1998] [Olsina, 1999]:

1. Nivel Conceptual (Goal): un objetivo/meta es definido para un propósito específico en base a las necesidades del proyecto u organización, teniendo en cuenta una variedad de razones, desde distintos puntos de vista relacionados a un ambiente en particular. Un objetivo/meta representa el nivel máximo de característica de calidad.
2. Nivel Operacional (Question): es un conjunto de preguntas que son utilizadas para caracterizar la forma de realización de una meta específica. Cada característica de nivel máximo es redefinida en las sub-características usando un conjunto de preguntas.
3. Nivel Cuantitativo (Metric)*:* es un conjunto de datos que está asociado a toda pregunta de manera cuantitativa. Para cuantificar una sub-característica se utiliza un conjunto de métricas. La interpretación de las métricas es utilizada para responder a las preguntas y concluir si la meta u objetivo se ha cumplido.

Asimismo se puede hacer un resumen de pasos a seguir para llevar a cabo el modelo y se representan en forma de flujo de actividades en la Figura 1.3 [Palacio, 2006] [Olsina, 1999]. Lo primero que se debe hacer es establecer los ***objetivos***/expectativas de la colección de datos, luego se desarrolla una lista de ***requisitos***/criterios de interés y se establecen los ***indicadores***. Seguidamente se diseñan los medios para recoger los datos (herramienta). Por último se recogen y validan los datos para analizarlos y dar las conclusiones finales.

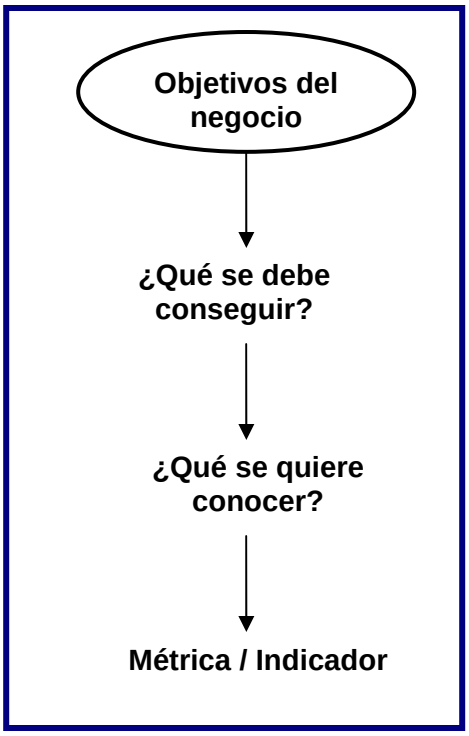

Figura 1. 3 Flujo de Actividades para el Paradigma GQM

1.4.3 Marco ISO/IEC 9126

La ISO/IEC 9126 es un estándar internacional para la evaluación del software. Es supervisado por el proyecto SquaRE (Security Quality Requirements Engineering - Ingeniería de Requisitos de Calidad de Seguridad) y la ISO 25000:2005, que siguen los mismos conceptos generales.

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.

Este estándar proviene del modelo establecido en 1977 por McCall y sus colegas, que propusieron un modelo para especificar la calidad del software. A pesar de que es uno de los más antiguos se ha extendido en todo el mundo y de él han derivado además del que se analiza en esta sección, muchos otros como el de Boehm (1978) o el SQM (Software Quality Management) de Murine (1988) [de Antonio, 2002].

1.4.3.1 Marco del modelo de calidad

El modelo de calidad que describe para la calidad de los productos software se divide en dos partes: a) calidad externa y calidad interna, b) 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 internos 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 1.4). 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].

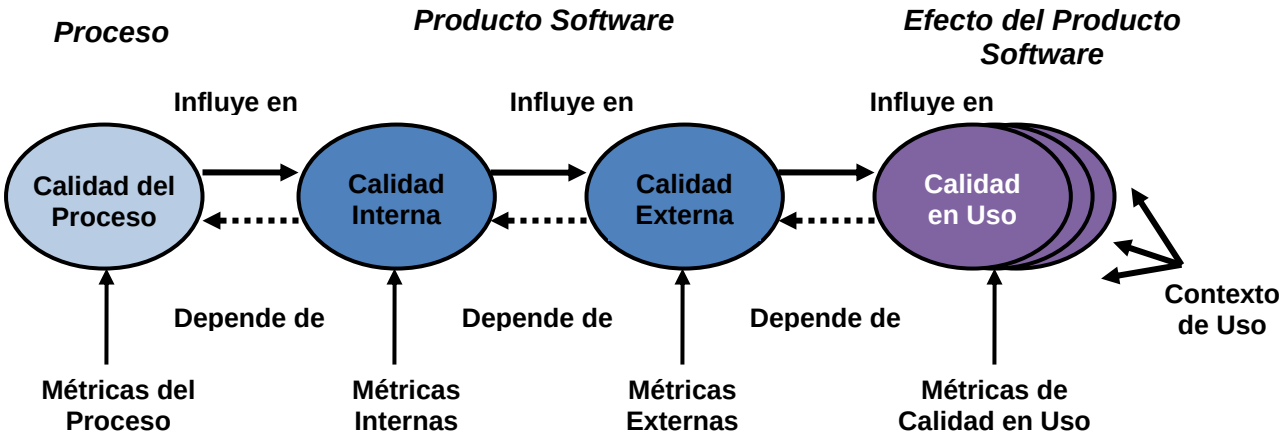

Figura 1. 4 Relación entre los diferentes enfoques hacia la calidad

Como muestra la Figura 1.4, las métricas internas pueden ser aplicadas a los productos intermedios que se desarrollan a lo largo del ciclo de vida de desarrollo de un producto software, tales como solicitud de propuesta, especificación de requisitos, especificaciones de diseño o código fuente. Las métricas internas 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. Esto le permite a los desarrolladores identificar los problemas que afecten la calidad e iniciar las acciones correctivas en las etapas tempranas del ciclo de vida de desarrollo del producto [ISO/IEC 9126-2, 2003].

Por su lado, las métricas externas pueden ser usadas para medir la calidad del producto software a través de la medición del comportamiento del sistema del cual el software forma parte (Figura 1.4). Las métricas externas solo pueden ser usadas durante las etapas de pruebas del proceso ciclo de vida y durante cualquier otra etapa operacional [ISO/IEC 9126-2, 2003].

Por último, las métricas de calidad en uso (Figura 1.4) 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 [ISO/IEC 9126-2, 2003].

Las necesidades de calidad del usuario pueden ser especificadas como requisitos de calidad a través de las métricas de calidad en uso, las métricas externas y algunas veces las métricas internas. Estos requisitos especificados por las métricas deben ser usados como criterios cuando el producto se evalúa [ISO/IEC 9126-2, 2003].

Es recomendable usar métricas internas que tengan una relación lo más fuerte posible con los objetivos de las métricas externas, así ellas pueden ser usadas para predecir los valores de las métricas externas. Sin embargo, a menudo es difícil diseñar un modelo teórico riguroso que proporcione una relación fuerte entre las métricas internas y externas [ISO/IEC 9126-2, 2003].

1.4.3.2 Utilización del modelo de calidad

La calidad del producto software se debe evaluar usando un modelo de calidad definido. El modelo de calidad se usará al fijar los objetivos de calidad para los productos del software (ejecutables) y productos software intermedios (especificaciones, diseño). La calidad del producto software se debe desglosar jerárquicamente en un modelo de calidad compuesto por características y sub-características, como se muestra en la Figura 1.5 que puede usarse como una lista de chequeo de problemas relacionados con la

calidad [NC-ISO/IEC 9126-1, 2005]. Las secciones posteriores definen un modelo jerárquico de la calidad, aunque pueden ser más apropiadas en circunstancias particulares otras vías de categorización de la calidad.

1.4.3.3 Modelo para la calidad interna y externa

Este modelo se ha desarrollado en un intento de identificar los atributos más importantes para la calidad interna y externa en un producto software. El modelo identifica seis características claves de calidad [NC-ISO/IEC 9126-1, 2005] donde cada una de ellas se descomponen en un conjunto de sub-características como se muestra en la Figura 1.5.

El significado de cada una de las características presentes en el modelo se muestra en el Anexo 1.

Los factores ISO/IEC 9126 no necesariamente se utilizan para medidas directas. En cualquier caso, facilitan una valiosa base para medidas indirectas y una excelente lista para determinar la calidad de un sistema.

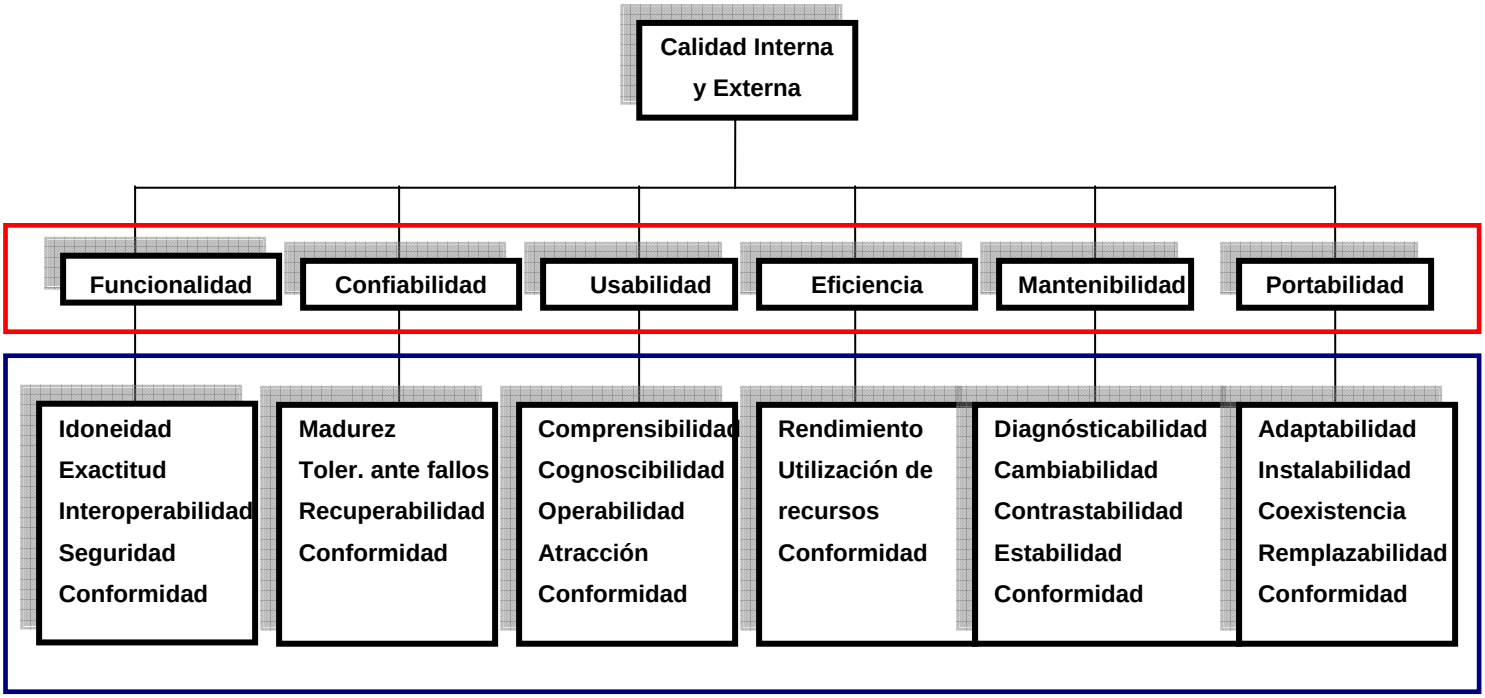

Figura 1. 5 Modelo ISO/IEC 9126 para la calidad interna y externa

1.5 Gestión de Requisitos

Desde el inicio del desarrollo de sistemas, los ingenieros se encuentran con un gran problema, la identificación de los requisitos del sistema. Esto es debido a que no es un proceso que pueda ser determinado matemáticamente. Es un proceso en el cual los datos son extraídos de las personas y estos datos pueden variar, dependiendo de la persona a la cual estemos consultando.

Por una parte, la definición de las necesidades del sistema es un proceso complejo [Letelier, 2003], pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

Por otro lado, teniendo en cuenta que el proceso de desarrollo de un software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente, existe una inmensa combinación de factores que impiden una verificación exhaustiva de todas las posibles situaciones que se puedan presentar (entradas, valores de variables, datos almacenados, software del sistema, otras aplicaciones que intervienen, el hardware sobre el cual se ejecuta, entre otros), además de lo intangible y abstracto que resulta ser un producto software, se hace aún más difícil la definición del producto y sus requisitos, sobre todo cuando no se tienen precedentes en productos software similares [Letelier, 2003].

Esto hace que los requisitos sean difíciles de consolidar en etapas tempranas del ciclo de desarrollo del software. Por lo que los cambios en los requisitos son inevitables, no sólo después de entregado el producto sino también durante el proceso de desarrollo.

Es por todo esto que la Gestión de Requisitos [Letelier, 2003] ha cobrado un papel fundamental en la Ingeniería de Requisitos (IR) y esta a su vez en la Ingeniería de Software trabajando arduamente para tratar de desarrollar técnicas que permitan hacer este proceso de una forma más eficiente, segura y sobre todo que garantice la calidad del resultado; en este intento se desarrolla el procedimiento propuesto que se abordará en el capítulo siguiente. De ahí la importancia de dedicarle todo el esfuerzo que sea posible a este proceso sin escatimar recursos y gastos porque esto preverá altos riesgos de errores en etapas posteriores.

1.5.1 Definiciones previas

A continuación, sería interesante presentar diferentes definiciones que han expresado indistintos autores del mundo de la Ingeniería de Software acerca del proceso que se estará tratando en este epígrafe con el propósito de entender los beneficios que proporcionaría una buena Especificación de Requisitos.

1.5.1.1 Gestión de Requisitos

Los autores Suzanne y James Robertson exponen que la *Gestión de Requisitos (GR)* es una exploración exhaustiva del producto deseado con el objetivo de descubrir y en algunos casos ingeniar la funcionalidad y el comportamiento de un producto. El resultado de este proceso es una ***especificación*** que sirve como entrada al análisis del sistema y posteriormente al diseño del producto [Cruz, 2005].

Por otra parte, el Modelo de Madurez de las Capacidades (Capability Maturity Model - CMM) plantea que el propósito de la *Gestión de Requisitos* es establecer un entendimiento común entre el usuario y los desarrolladores en cuanto a los requisitos del usuario que serán abordados por el proyecto de software [Cruz, 2005].

Por su parte, Roger S. Pressman afirma que la Gestión de Requisitos es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento [Pressman, 2002], de igual modo, lo describe como: *"… un proceso de descubrimiento, refinamiento, modelado y* ***especificación****"*. Además afirma que tanto el cliente como el desarrollador tienen un papel activo en la gestión de requisitos del software, donde el cliente intenta replantear un sistema confuso a nivel de descripción de datos, funciones y comportamiento; el desarrollador en cambio actúa como interrogador, como consultor, como persona que resuelve problemas y como negociador [Cruz, 2005].

Es interesante destacar algunos aspectos comunes que se reflejan en las definiciones anteriores, como la importancia que se le otorga a lograr una *comunicación* fluida entre clientes/usuarios y desarrolladores, además de alcanzar una buena *especificación* por la relación estrecha que mantiene con procesos posteriores, sirviéndole de base.

1.5.1.2 Captura de Requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa [Escalona & Koch, 2002]. Alguna de estas técnicas son: Entrevistas, JAD (Joint Application Development - Desarrollo conjunto de aplicaciones), Brainstorming (Tormenta de ideas), Casos de Uso, ***Cuestionarios*** y/o ***Checklists (Listas de Chequeo)*** (Epígrafe 1.2.3.3), entre otras, pero no se tratarán específicamente cada una de ellas porque están fuera de los objetivos de esta investigación [Escalona & Koch, 2002], a excepción de las Listas de Chequeos que se hará uso de ellas para la especificación.

1.5.2 Problemas de la Gestión de Requisitos

En el proceso de Gestión de Requisitos se pueden manifestar diversos problemas que son fundamentados en [Durán, 2000] [Cruz, 2005] y los clasifican en cinco grandes grupos que se describen brevemente a continuación:

- ♠ ***Problemas de articulación***

Los problemas de articulación están relacionados con la expresión de las necesidades por parte de clientes y usuarios, y la comprensión de dichas necesidades por parte de los desarrolladores. Por ejemplo, los clientes y usuarios pueden no ser capaces de expresar sus necesidades apropiadamente, algunos usuarios pueden no expresar sus necesidades por miedo a parecer incompetentes ante los demás, también puede suceder que algunos desarrolladores no escuchen apropiadamente a los clientes y usuarios, porque creen haber entendido sus necesidades rápidamente, o porque se dedican a pensar inmediatamente sobre aspectos de implementación y no se ponen en el lugar de clientes y usuarios.

- ♠ ***Problemas de comunicación***

Las dificultades en la comunicación entre clientes, usuarios y desarrolladores pueden ser ocasionadas porque los clientes, usuarios y desarrolladores tienen culturas y vocabularios diferentes, o porque el medio

de comunicación que es utilizado puede no ser entendible por todos los participantes, el lenguaje natural es el que más se utiliza porque es el único medio de comunicación común a todos los participantes, pues la utilización de otro tipo de técnicas como diagramas o lenguajes artificiales puede presentar problemas de compresión. El analista debe ser capaz de comunicarse y tratar con todo tipo de personas y ser capaz de manejar conflictos personales y políticos.

- ***Problemas de limitaciones cognitivas***

Los problemas de las limitaciones cognitivas del ser humano aparecen en las actividades de obtención de requisitos, pues el ingeniero de requisitos (analista) debe tener un conocimiento adecuado del dominio del problema y no hacer suposiciones sobre ello, al igual que los clientes y usuarios no deben hacer suposiciones sobre aspectos tecnológicos. El analista debe ser capaz de manejar problemas complejos y asegurarse de que todos los temas importantes sean tratados.

- ***Problemas de conducta humana***

La naturaleza social de la obtención de requisitos provoca que surjan problemas de conducta humana, pues puede haber conflictos y ambigüedades en los roles que cada persona debe jugar en el proceso de obtención. La suposición o el temor a que el sistema a desarrollar cambie su forma de trabajar o incluso ponga en peligro su puesto de trabajo, pueden provocar que algunos usuarios retengan información o incluso saboteen el desarrollo, por ejemplo proporcionando información falsa.

- ***Problemas técnicos***

Otros problemas de la obtención de requisitos pueden considerarse como *técnicos.* El software tiene que resolver problemas cada vez más complejos, por lo que sus requisitos son también cada vez más complejos y contemplan detalles cada vez más específicos del dominio del problema. Los requisitos cambian en el tiempo y también el hardware y el software cambian rápidamente, haciendo asequibles requisitos que antes eran inabordables por su complejidad o por su costo. Puede haber muchas fuentes de requisitos, por lo que a veces no basta con consultar con los clientes y usuarios, sino que también es necesario hacerlo con técnicos, personal de mantenimiento, consultar normativas, estándares, entre otros.

1.5.3 Proceso de Gestión de Requisitos

Existen diferentes Modelos que representan a la Gestión de Requisitos, creados por diferentes autores como Suzanne y James Robertson, Pohl, Sommerville y Sawyer, Amador Durán, entre otros [Durán, 2000] [Cruz, 2005]. Todos estos modelos encierran un conjunto de procesos y actividades similares para garantizar que se desarrolle el software con una alta calidad.

En este apartado se va a presentar el modelo que fue creado por el proyecto SWEBOK (Software Engineering Body of Knowledge) de la IEEE y la ACM para producir un cuerpo de conocimiento sobre ingeniería de software que siente las bases de dicha ingeniería como una profesión [Durán, 2000]. Dicho modelo lleva el mismo nombre del proyecto y se escogió por la claridad de los procesos que describe (Figura 1.6) [Durán, 2000].

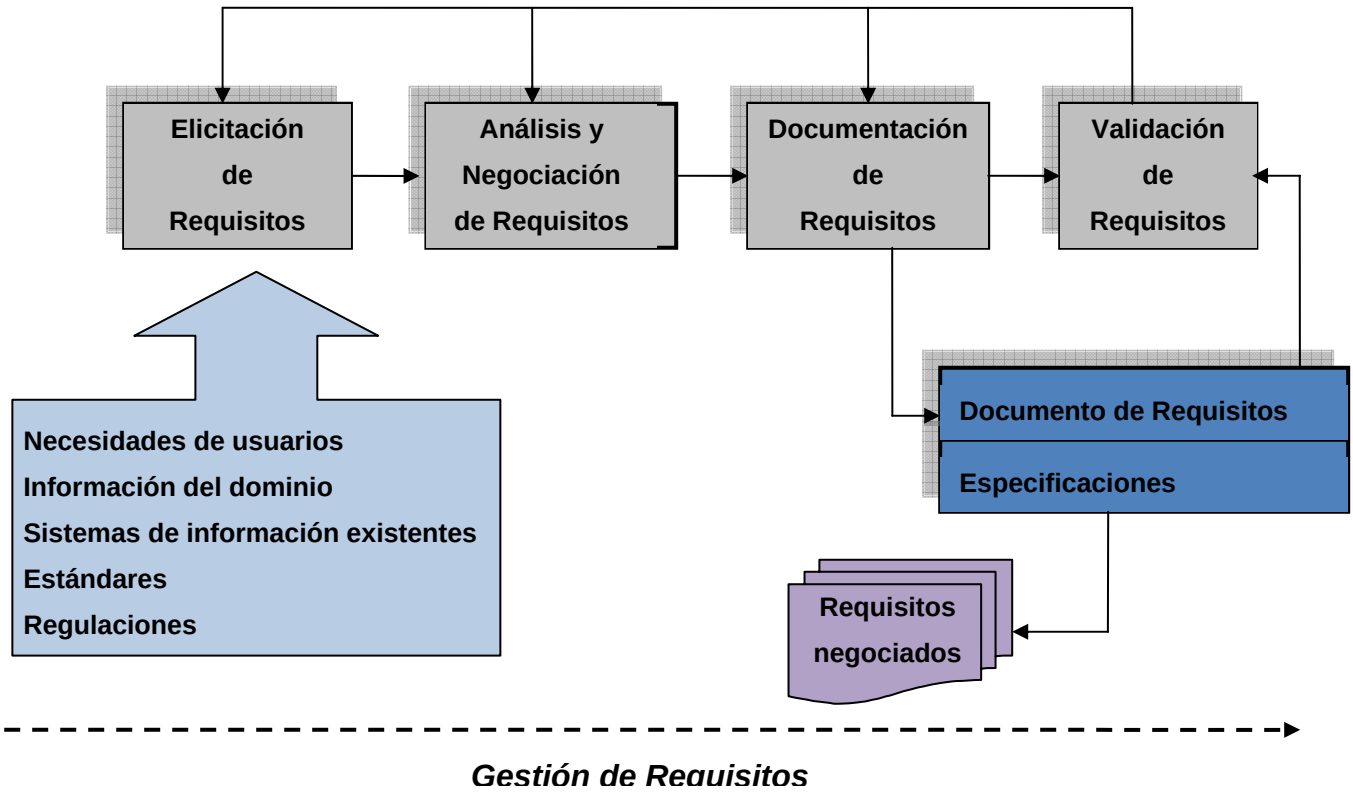

Figura 1. 6 Modelo SWEBOK del proceso Gestión de Requisitos

La Figura 1.6 representa las actividades básicas que propone el modelo presentado, sus relaciones y los artefactos que se obtienen al finalizar el proceso. La primera actividad se dirige a la recopilación de los requisitos y se considera que es una actividad meramente humana; en la segunda actividad se detectan los principales problemas y se negocian los conflictos encontrados; en la siguiente actividad básicamente se define la forma de documentación apropiada; luego se comprueban los requisitos usando las listas de chequeo para validar y verificar la calidad de los mismos; finalmente la gestión de requisitos se extiende a todas las actividades anteriores y principalmente vela por los cambios y mantenimiento de los requisitos. Los artefactos más importantes de este proceso lo constituyen las especificaciones.

Sin embargo, dentro de todas las actividades que se describieron anteriormente no se hace referencia a las actividades de SQA que se pueden realizar para cada una de ellas.

Una explicación más detallada de estas actividades se presenta en el Anexo 2.

1.5.4 Especificación de Requisitos del Software

La Especificación de Requisitos del 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].

La especificación, independientemente de la forma en que se realice, puede verse como un proceso de representación. Se centra en la colección y la organización de todos los *requisitos* que rodean el proyecto. Hace referencia al plan de Gestión de Requisitos para determinar la localización y organización correctas de los requisitos. Por ejemplo, se puede desear tener una ERS separada para describir los requisitos completos del software para cada característica en un comportamiento particular del producto. Esto puede incluir varios *casos de uso* del *modelo de casos de uso* del sistema, para describir los *requisitos funcionales* de esta característica, junto con el sistema relevante de requisitos detallados en *especificaciones suplementarias* [RUP, 2003].

Puesto que puede ser que los desarrolladores se encuentren con diversas herramientas para recolectar estos requisitos, es importante realizar la colección de requisitos teniendo en cuenta varios artefactos y

diversas herramientas. Por ejemplo, puede ser que encuentre apropiado recoger requisitos textuales por ejemplo los requisitos no funcionales, con una herramienta de documentación textual en especificaciones suplementarias. Por otra parte, puede ser que encuentre útil recoger algunos (o todos) los requisitos funcionales en los casos de uso y puede ser que encuentre práctico utilizar una herramienta apropiada a las necesidades de definir el modelo del casos de uso [RUP, 2003].

La importancia de una buena especificación puede evaluarse por los beneficios específicos que provee a todas las personas que intervienen en el proceso de desarrollo de un software (clientes/usuarios, desarrolladores y otros involucrados). Dado que permite establecer las bases de un acuerdo entre clientes/usuarios y desarrolladores en cuanto a lo que el producto software debe hacer, reduce el esfuerzo desarrollado para evitar rediseños, recodificación y repruebas, provee las bases para la estimación del costo y entrega del producto, así como una línea base para la validación y verificación, facilita la transferencia a usuarios nuevos o máquinas nuevas y sirve como una base para el aumento del producto final.

No cabe duda de que el modo de especificación tiene mucho que ver con la calidad de la solución. Muchos ingenieros se han visto forzados a lidiar con especificaciones incompletas, inconsistentes o engañosas y han experimentado la frustración y confusión que invariablemente provocan. La calidad, la fecha de entrega y el alcance del software son las que sufren las consecuencias [Pressman, 2002].

En el Anexo 3 se presenta un prototipo de las partes que debe contener una buena Especificación.

1.5.5 Calidad de la Especificación de Requisitos del Software

De acuerdo con el estándar [IEEE 830, 1998], se considera que una especificación (Software Requirements Specification, (SRS-ERS)) es de “calidad” cuando se puede decir de ella que es “correcta, no-ambigua, completa, consistente, ordenada por importancia y/o estabilidad, verificable, modificable y trazable”. A continuación se detalla cada una de ellas [IEEE 830, 1998] [Monzón, 2002].

1.5.5.1 Especificación correcta

Una definición clásica de corrección indica que “un conjunto de requisitos software es correcto sólo si todos los requisitos contenidos representan algo que es requerido para la construcción del sistema y no hay errores que afecten al diseño”

Un aspecto fundamental a tener en cuenta acerca de la corrección es que depende del usuario final del sistema, por lo tanto es quien decide si la ERS es correcta o no. Un problema habitual en este ámbito es la omisión (usuarios) y adición (analistas) de información.

1.5.5.2 Especificación no-ambigua

Se considera que un requisito individual, como ítem de información, es ambiguo cuando puede ser interpretado de formas diversas por diferentes personas. Así, el estándar *IEEE 830* establece que un requisito es no-ambiguo “si, y sólo si, puede estar sujeto a una única interpretación.”

Asegurar que una ERS está libre de ambigüedades es difícil, pues las personas que intervienen en este proceso provienen de diferentes culturas y tienen sus propios enfoques de percibir los conceptos.

1.5.5.3 Especificación completa

El estándar establece que una especificación es completa “si, y sólo si, describe todos los requisitos relevantes para el usuario, incluyendo requisitos asociados con funcionalidad, actuación, restricciones de diseño, atributos o interfaces externas.”
La persona a cargo de este proceso debe tener toda la información necesaria para especificar hasta el más mínimo detalle del sistema. Un problema en este sentido es la falta de un mecanismo que permita detectar la falta de requisitos necesarios para completar el sistema.

1.5.5.4 Especificación consistente

Se considera que una especificación es consistente “si, y sólo si, no hay ningún subconjunto de requisitos descrito dentro de ella que esté en conflicto con cualquier otro.”

El mayor problema en este ámbito es la repetición de información en los requisitos (requisitos repetitivos y redundantes). A veces se usa la misma palabra para designar conceptos diferentes (incoherencias).

1.5.5.5 Especificación ordenada por importancia y/o estabilidad (organizada)

La categorización por importancia permite establecer prioridades a la hora de abordar el desarrollo. Categorizar los requisitos por atributo de "importancia" o "prioridad" es muy práctico, dado que la construcción de sistemas se hace con un presupuesto y plazo limitados.

Por otro lado, la categorización por estabilidad es importante porque el cambio de los requisitos de usuarios es intrínseco al propio cambio en el sistema. Las necesidades del mercado que se imponen también influyen.

1.5.5.6 Especificación verificable

Se considera que una especificación es verificable "si lo son cada uno de los requisitos constituyentes". A su vez, se considera que un requisito individual es verificable "si existe un proceso acotado (en plazo y presupuesto) que permita determinar que el sistema construido satisface lo descrito en el propio requisito".

La descripción detallada y prueba de los requisitos una vez implementados ayudan considerablemente a su verificación.

1.5.5.7 Especificación modificable

Se considera que una especificación es modificable "si su estructura es tal que permite realizar cambios sobre los requisitos que contiene de forma sencilla, completa y consistente, manteniendo la estructura inicial del conjunto".
La gestión de configuración juega un papel fundamental en este sentido; además de una buena organización de la información y lograr un bajo acoplamiento entre requisitos.

1.5.5.8 Especificación trazable

Una especificación se considera trazable si el origen de cada requisito individual está claro y existe algún mecanismo que permita seguir el impacto de dicho requisito a lo largo del resto de actividades del ciclo productivo.

Lo más importante para este aspecto es que todos los requisitos deben estar relacionados, al menos, con un requisito de usuario.

Por otro lado, el autor Alan Davis [Sumano, 2000] coincide con las características anteriores aunque agrega otras que considera necesario estén presentes en una Especificación de Requisitos de Software:

1. *Entendible por el cliente*: La ERS debe escribirse evitando en lo posible los formalismos. Aunque la falta de formalismos puede llevar a ambigüedades y en ocasiones es necesario incluirlos, se debe tener en mente que quien leerá la ERS es, en la mayoría de los casos, una persona no especialista en computación.
2. *Trazable hacia atrás*: Una ERS es trazable hacia atrás si para cada requisito se sabe su origen; esto es, existe un soporte (persona o documento) que lo originó y que pertenece a las etapas de desarrollo anteriores.
3. *Trazable hacia delante*: Una ERS es trazable hacia delante si está escrita de tal forma que se facilita la referencia a cada requisito individual por las actividades de las etapas posteriores (diseño, prueba).
4. *Independiente del diseño*: Una ERS es independiente del diseño si no involucra arquitectura de software específica o algoritmos.
5. *Anotada*: Una ERS debe asignar a cada requisito tanto su necesidad relativa como su estabilidad relativa dentro del sistema. Con tales anotaciones para cada requisito se dará prioridad al cumplimiento de ellos respecto a los tiempos de entrega.
6. *Concisa*: Dadas dos ERS del mismo sistema en que cada una contiene los mismos elementos de calidad anteriores, la más breve es la mejor.
7. *Organizada*: Una ERS es organizada si los requisitos contenidos en ella son fáciles de localizar. Esto involucra el hecho de que los requisitos que están agrupados deben estar cerca entre sí.

Los dos puntos de vistas propuestos por diferentes autores tienen muchos aspectos en común, por lo que se considera que una especificación que cumpla con las características mencionadas anteriormente presenta la calidad requerida.

1.6 Métricas del Software

En este apartado se estarán abordando temas referentes a las métricas haciendo énfasis en las métricas de calidad que se van a aplicar a la ERS y van a permitir medirla y evaluarla cuantitativamente.

1.6.1 Necesidad de las métricas

Las instituciones que trabajan con proyectos de software están sometidas a las presiones y desafíos del mercado por lo que la gestión de proyectos y mejora de la calidad son actividades fundamentales para su supervivencia. A pesar de que la Ingeniería de Software ha introducido y popularizado una serie de estándares para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí, y además ha surgido un número creciente de herramientas automatizadas para ayudar a definir y aplicar un proceso de desarrollo de software efectivo se agudiza una alta incidencia de fallos en los proyectos de software. Por esto se ve parte de la solución en la utilización de métricas durante todo el ciclo de desarrollo del software enfatizando en las primeras etapas a las que menos se le ha dedicado en lo que a ello respecta [Andújar, 2004].

Según Park, Goethert y Florac [Pressman, 2002] existen cuatro razones fundamentales por las que es necesario realizar mediciones, tanto al proceso como al producto software y sus recursos. Estas son: caracterizar (1) para comprender mejor los procesos, productos, recursos y entornos, además para establecer las líneas base para comparaciones con evaluaciones futuras; evaluar (2) para conocer cuándo los proyectos y procesos andan por el camino equivocado, de modo que se puedan poner bajo control, además para valorar la consecución de los objetivos de calidad y evaluar las mejoras del proceso en los productos; predecir (3) para hacer planificaciones, que significa aumentar la comprensión de las relaciones de los procesos, productos y construcción de modelos de estas relaciones, así los valores que se observan para algunos atributos pueden utilizarse para predecir otros, además las medidas de predicción son la base para la estimación de costos, tiempo, calidad, análisis de riesgos y realización de cambios; por último mejorar (4) la calidad del producto y rendimiento del proceso cuando se ha recogido información cuantitativa que ayuda a identificar obstáculos, problemas de raíz e ineficiencias.

1.6.2 Objetivos de las métricas

Las métricas tienen dos objetivos bien definidos, primero establecer un conjunto abierto de procedimientos

e indicadores adecuados de calidad y segundo definir métodos para medir los indicadores de calidad del software, es este contexto es verdaderamente significativo que las características de calidad puedan ser cuantificables y medibles.

El ciclo de vida del software es el conjunto de fases y actividades por las que atraviesa hasta que finaliza [Andújar, 2004]. Tenerlo en cuenta es fundamental para el estudio de las métricas, ya que éstas pueden aplicarse a cualquier tarea, actividad o fase del producto o proceso, por ello es tan importante conocer a profundidad el proceso que se pretende evaluar a través de las métricas.

Lograr un software de buena calidad no solo requiere considerar algunas fases del ciclo de vida. En cada una de ellas existen contribuciones decisivas para la calidad. Un error en una de las fases iniciales puede ocasionar inmensos problemas en las posteriores lo que supone un considerable incremento de los costos a la vez que la probabilidad de un deterioro de la calidad o aumento del tiempo de salida del producto es muy alta [Andújar, 2004]. Por esta razón se debe cubrir todas las fases del ciclo de vida del software con el propósito de lograr hacer las cosas bien desde el principio, tanto para la Gestión de requisitos como para el análisis de los requisitos, diseño de la arquitectura, implementación y pruebas [Andújar, 2004].

1.6.3 Medidas, Medición, Métricas e Indicadores

A pesar de que los términos medida, medición y métricas se utilizan con frecuencia indistintamente, es importante destacar las diferencias sutiles entre ellos. Dentro del contexto de la ingeniería de software, una medida proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. La medición es el acto y efecto de determinar una medida [Pressman, 2002] [Andújar, 2004], mientras que Norman E. Fenton la define como el proceso mediante el cual se le asignan números o símbolos a atributos de entidades del mundo real de tal manera que las describan de acuerdo a reglas claramente definidas [Fuente, 2001] [Aguillón, 2007]. El IEEE Standard Glossary of Software Engineering Terms define métrica como “una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado” [IEEE 729, 1983], por su parte en [Barceló, 2003] se define como la asignación de valor a un atributo[1] de una entidad propia del software, ya

[1] Cuando se habla de un ***atributo*** se refiere a una determinada característica que se pretende medir y cuantificar (como puede ser el esfuerzo, la fiabilidad, la calidad, el tamaño).

ya sea un producto[1] o un proceso[2], este concepto se ajusta mejor a los objetivos de esta investigación.

Se debe destacar que las mediciones del software pueden clasificarse en *medidas directas* y *medidas indirectas* [Pressman, 2002] [Perez, 2004].

Entre las medidas directas del proceso de la ingeniería de software se incluyen el costo y el esfuerzo aplicados. Entre las medidas directas del producto se encuentran las líneas de código (LCD) producidas, velocidad de ejecución, tamaño de memoria y los defectos informados durante un período de tiempo establecido [Pressman, 2002] [Perez, 2004]. Mientras que entre las medidas indirectas se incluyen la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento y otras "capacidades". De esto último se estará tratando más al detalle posteriormente en este documento [Perez, 2004] [Pressman, 2002].

Por una parte, Pressman plantea que un ingeniero del software recopila medidas y desarrolla métricas para obtener indicadores. Un *indicador* es una métrica o combinación de ellas que proporcionan una visión profunda del proceso del software, del proyecto de software o del producto en sí. Un indicador proporciona una visión profunda que permite al gestor de proyectos o a los ingenieros de software ajustar el producto, el proyecto o el proceso para que las cosas salgan mejor [Pressman, 2002].

1.6.4 Proceso de medición

Las métricas, según Pressman, deben ayudar a la (1) evaluación de los modelos de análisis y de diseño, (2) donde proporcionen una indicación de la complejidad de diseños procedimentales y de código fuente, y (3) ayuden en el diseño de pruebas más efectivas; por lo que propone un proceso de medición que se caracteriza en cinco actividades.

La primera actividad es la *formulación*, donde se obtienen las medidas y métricas apropiadas para la representación del software en cuestión. Sigue la *colección*, como mecanismo empleado para acumular los datos necesarios para obtener las métricas formuladas. Luego se realiza un *análisis* del cálculo de las métricas y si se puede se aplican herramientas matemáticas. Después se hace una *interpretación*,

[1] Cuando se habla de un ***producto*** se refiere, por ejemplo, a un código, un diseño o una especificación, etc.
[2] Cuando se habla de un ***proceso*** se hace referencia a una etapa de la construcción del software y a la manera de llevarlo a cabo: requisitos, diseños, pruebas, etc.

evaluando los resultados de las métricas en un esfuerzo por lograr la calidad interna y se termina con una *retroalimentación* (feedback), donde las recomendaciones obtenidas de la interpretación de las métricas se transmiten al equipo que trabaja en la construcción del software [Pressman, 2002].

1.6.5 Métricas del Software

Paul Goodman en su libro *Software Metrics-Best Practices for Successful IT Management* [Goodman, 2004] [Rossi, 2003] [González, 2001] define las métricas de software como: "La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos para suministrar información relevante a tiempo, así el desarrollador junto con el empleo de estás técnicas mejorará el proceso y sus productos". Las métricas de software proveen la información necesaria para la toma de decisiones técnicas [González, 2001] [Goodman, 2004].

Por su parte, el Dr. Andújar [Andújar, 2004] agrega que el propósito de las métricas de software es permitir evaluaciones a través del ciclo de vida del software para asegurar que se alcanzan los requisitos del producto y de la organización.

Las métricas del software ayudan a [Andújar, 2004]:

- Seguir objetivamente a los proyectos, tendiendo en cuenta impactos en el cumplimiento de plazos, costos y calidad (satisfacción del cliente).
- Estimar y predecir tendencias en función de la organización, los procesos, la gestión del cambio, la gestión de la configuración, medidas de software, etc.
- Evaluar de la productividad.
- Tomar decisiones objetivas.
- Comunicarse a través de un lenguaje común.
- Identificar los puntos críticos del proyecto para garantizarlos.
- Gestionar los riesgos.
- Identificar las mejores prácticas por comparación objetiva de métricas.
- Identificar y eliminar lo antes posible las causas de los principales defectos.
- Mejorar el entendimiento del proceso.

- ♠ Mejorar el proceso software mediante una estrategia a medio y largo plazo.

Al mismo tiempo Andújar afirma que medir con precisión es complejo por lo que necesita apoyarse en ciertas características para que realmente sirva como indicador [Andújar, 2004]. Algunas de estas características son la validez de criterio que debe servir para medir lo que se desea controlar, la adecuación, que no es más que la conformidad a lo que se quiere medir, por último la consistencia que es el grado de mantener su valor en el tiempo y en las mismas condiciones.

1.6.6 Métricas en el proceso y el producto

Las métricas del proceso se coleccionan de todos los proyectos y durante un extenso período de tiempo con el objetivo de proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo [Pressman, 2002]. En algunos casos, se pueden utilizar las mismas métricas del software para determinar tanto el proyecto como los indicadores del proceso.

Según Pressman la forma más óptima de mejorar cualquier proceso es midiendo los atributos del mismo, desarrollando un juego de métricas significativas a partir de estos atributos y luego utilizar las métricas para proporcionar indicadores que conducirán a una estrategia de mejora. Es importante destacar que el proceso es el único factor de "los controlables al mejorar la calidad del software y su rendimiento como organización" [Pressman, 2002].

Por una parte, es importante determinar cómo se puede medir la efectividad de un proceso de software. La eficacia de un proceso de software se mide indirectamente, es decir, se extrae un conjunto de métricas según los resultados que provienen del proceso. Entre estos resultados se encuentran medidas de errores detectados antes de la entrega del software, defectos detectados e informados a los usuarios finales, productos de trabajos entregados (productividad), esfuerzo humano y tiempo consumido, ajuste con la planificación y otras medidas [Pressman, 2002] [Rossi, 2003]. Las métricas de proceso también se extraen midiendo las características de tareas específicas de la ingeniería de software [Pressman, 2002].

Por otro lado, Grady argumenta que existen usos "privados y públicos" para distintos tipos de datos de proceso. Las métricas recopiladas sobre una base particular deben ser *privadas* para el individuo y utilizarse como un indicador de ese individuo. Ejemplos de métricas privadas pueden ser el índice de

defectos (individualmente), índices de defectos (por módulos), errores encontrados durante el desarrollo [Pressman, 2002].

La filosofía de “datos de proceso privado” se ajusta muy bien con el enfoque del *Proceso Personal del Software (PSP)* propuesto por Humphrey. Él reconoce que la mejora del proceso del software puede y debe empezar en el nivel individual. Los datos privados de proceso pueden servir como referencia importante para mejorar el trabajo individual del ingeniero del software [Humphrey, 2001].

Algunas métricas de proceso son privadas para el equipo del proyecto de software, sin embargo son *públicas* para todos los miembros del equipo. Entre los ejemplos se pueden citar los defectos informados de funciones importantes del software (que un grupo de profesionales han desarrollado), errores encontrados durante revisiones técnicas formales y líneas de código o puntos de función por módulo y función. El equipo revisa estos datos para detectar los indicadores que pueden mejorar el rendimiento del equipo [Humphrey, 2001].

Las métricas del proceso del software pueden proporcionar beneficios significativos a medida que una organización trabaja por elevar su nivel global de madurez del proceso. A medida que una organización está más conforme con la recopilación y utiliza métricas de proceso, la derivación de indicadores simples abre el camino hacia un enfoque más estricto llamado *Mejora Estadística de Proceso del Software (MEPS)* [González, 2001]. En esencia, MEPS utiliza el análisis de fallos del software para recopilar información de errores y defectos encontrados al desarrollar y utilizar una aplicación de sistema o producto. El análisis de fallos funciona de la siguiente manera [Pressman, 2002] [González, 2001]:

1. Todos los errores y defectos se categorizan por origen (ejemplo: defectos en la especificación, en la lógica, no acorde con los estándares).
2. Se registra tanto el costo de corregir cada error como el del defecto.
3. El número de errores y defectos de cada categoría se cuentan y se ordenan descendentemente.
4. Se computa el costo total de errores y defectos de cada categoría.
5. Los datos resultantes se analizan para detectar las categorías que producen el costo más alto para la organización.

6. Se desarrollan planes para modificar el proceso con la intención de eliminar (o reducir la frecuencia de apariciones de) la clase de errores y defectos que sean más costosos.

Por otro lado, las métricas del producto son mediciones del producto software. Esta definición incluye el tamaño del producto, la complejidad de la estructura lógica y la complejidad de estructuras de datos, entre otros [Rossi, 2003]. Además está estrechamente relacionada con las mediciones del proceso.

1.6.7 Métricas de Calidad en el Software

Para producir sistemas, aplicaciones o productos de alta calidad los ingenieros del software deben aplicar métodos efectivos junto con herramientas modernas dentro del contexto de un proceso maduro de desarrollo de software, esto incluye medir si la alta calidad se está llevando a cabo [Pressman, 2002] [Perez, 2004].

La calidad de un sistema, aplicación o producto es tan buena como los requisitos que describen el problema, el diseño que modela la solución, el código que conduce a un programa ejecutable y las pruebas que ejercitan el software para detectar errores, en este sentido los desarrolladores deben realizar y utilizar mediciones. Para alcanzar esta evaluación de la calidad en tiempo real, se deben utilizar medidas técnicas que evalúan la calidad con objetividad, no con subjetividad.

A pesar de que se pueden recolectar varias medidas de calidad, el primero de los objetivos en el proyecto es medir los errores y defectos [Rossi, 2003] [Pressman, 2002]. Las métricas que provienen de estas medidas proporcionan una indicación de la efectividad de las actividades de control y de aseguramiento de la calidad en grupos o en particulares.

1.7 Conclusiones del capítulo

Este capítulo se limitó a fundamentar toda una teoría acerca temas de calidad, requisitos y métricas. Hasta el momento se han analizado un conjunto de factores cualitativos para la “medición” de la calidad del software. La subjetividad y la especialización influyen en la determinación de la calidad y se necesita una definición de calidad de software más exacta, por lo que sería conveniente hacer una transición a una visión cuantitativa que permita realizar un análisis objetivo con el fin de evaluar la calidad. Aunque hay que tener en cuenta que no existe el conocimiento absoluto, por lo que no se debe esperar poder medir la

calidad del software exactamente, pues cada medición es parcialmente imperfecta [Pressman, 2002]. De esta transición se estará tratando en el próximo capítulo con una propuesta más práctica.

Capítulo 2. Procedimiento propuesto para medir la calidad en la Gestión de Requisitos y su especificación

2.1 Introducción

En este capítulo se presenta la propuesta del procedimiento RPQ (*Requirements Process Quality – Calidad en el Proceso Gestión de Requisitos*), donde se describen brevemente cada una de sus cuatro fases y se fundamentan dos procedimientos de medición, uno está enfocado a evaluar la calidad del proceso de Gestión de Requisitos basándose en el paradigma GQM, que se lleva a cabo a través de las revisiones que se realizan al proceso y sus respectivos elementos de especificación (EE). El segundo procedimiento se enfoca a la calidad del producto, particularmente a la Especificación de Requisitos del Software, se basa en la metodología propuesta en el estándar IEEE 1061 y está soportado por el marco ISO/IEC 9126 conjunto a las características de calidad de la especificación que propone el estándar IEEE 830. Para este caso se evalúa la calidad de la especificación como un todo y no por EE, con este propósito se utiliza una lista de chequeo.

2.2 Descripción general del Procedimiento RPQ

Este procedimiento propone un enfoque sistemático, disciplinado y cuantitativo que se adecua a la evaluación, comparación y análisis de calidad del proceso de Gestión de Requisitos y a su correspondiente Especificación de Requisitos del Software.

El procedimiento se caracteriza por ser integral, robusto y flexible, cubre la mayor parte de las actividades en el proceso de Gestión de Requisitos. Es integral porque permite a partir de metas u objetivos, requisitos y usuarios específicos medir atributos o sub-características de calidad y sacar conclusiones de la ERS. Se requiere un procedimiento integral para especificar el árbol de requisitos de características y atributos de calidad, para determinar los criterios de medición elementales e implementarlos, y para realizar la agregación apropiada de manera que se pueda producir un indicador global. Es decir, un conjunto bien definido y cooperante de estrategias, métodos, modelos y técnicas que, aplicados sistemáticamente a las distintas actividades del proceso, que contribuyen a obtener los indicadores esperados. Este procedimiento cubre las principales actividades del proceso de medición que se describió

en el epígrafe 1.6.4.

Pero además, el procedimiento es robusto por permitir procesos con resultados repetibles, reproducibles y objetivos. En cuanto a la característica de repetitibilidad de la medición de un mismo producto y para la misma especificación de requisitos y por los mismos evaluadores implica que debe producir resultados aceptados como idénticos. En cuanto a reproducibilidad de la medición de un mismo producto y para la misma especificación de requisitos y realizada por evaluadores diferentes implica que debe producir resultados aceptados como idénticos [ISO/IEC 14598-5]. En cuanto a objetividad, los resultados deben ser factibles, observables y cuantitativos, no desviados por la opinión parcial o intencional de los evaluadores.

Finalmente, el procedimiento es flexible. Primero, porque permite agregar o quitar características, sub-características y atributos de un modo modular (con la idea de cohesión y mínimo solapamiento), conforme a las necesidades específicas del perfil o perfiles de usuario y del contexto organizacional. Segundo, porque dependiendo del dominio, criticidad del artefacto y necesidad de precisión, permite ajustar los criterios de medición y evaluación. Tercero, es flexible en el sentido que dependiendo de la complejidad en la cantidad de componentes y/o relaciones, se pueden utilizar modelos más sencillos.

2.3 Panorama de las fases del procedimiento RPQ

En esta sección se describen las principales fases (pasos, actividades) que van a conformar al procedimiento propuesto, que se representa gráficamente en la Figura 2.1.

Las fases son las siguientes:

- ♠ Planificación y Programación de la Medición de Calidad
- ♠ Medición al Proceso
- ♠ Medición al Producto
- ♠ Análisis de Resultados, Conclusión y Documentación

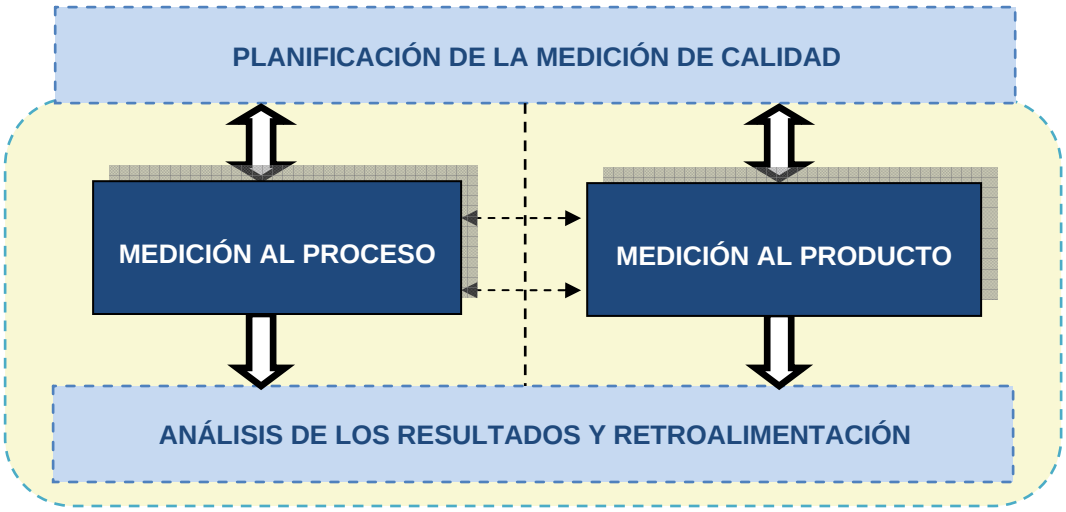

Figura 2. 1 Fases del procedimiento RPQ

La fase de *Planificación de la Medición de Calidad*, contiene actividades con el fin de determinar objetivos estratégicos, tácticos y operativos. Se establecen las principales estrategias y metas del proceso en un contexto organizacional; se define el proceso de medición, asignando métodos, agentes y recursos a las actividades; se programan y replanifican una vez en marcha el proceso de medición.

La fase de *Medición al Proceso* integra actividades, modelos, técnicas y herramientas para determinar métricas y criterios de medición para el proceso de Gestión de Requisitos. Se describe un procedimiento de medición orientado a metas con el fin de evaluar, comparar, analizar y mejorar atributos internos (tiempo, esfuerzo, número de cambio, entre otros) del proceso que deben responder a necesidades del proyecto y la organización. Una vez definidos los criterios para medir el proceso. El procedimiento propone ejecutar un proceso de recolección de datos que se realiza a través de revisiones, luego calcular las métricas, analizar y documentar los resultados. Se propone en esta fase utilizar herramientas para automatizar las actividades mencionadas anteriormente. En la descripción de esta fase se discute detalladamente cada una de sus actividades.

Durante la fase de *Medición al Producto* se incluyen actividades, modelos, técnicas y herramientas para determinar la calidad de la especificación de requisitos como producto obtenido en el proceso de Gestión

de Requisitos. Se propone un procedimiento de medición que comienza con la definición y especificación de características y sub-características de calidad, utilizando la clasificación de la ISO/IEC 9126 (funcionalidad, usabilidad, y otras), incorporando un conjunto de actividades para la determinación y análisis de las mismas. El proceso de determinación de características y sub-características termina con un documento que especifica a todas las características y sub-características que se pueden aplicar a la especificación y finalmente se propone un conjunto de métricas para calcular el grado de presencia de estas características en la especificación. En la descripción de esta fase se amplían los principales elementos que la conforman.

Por último, la fase de *Análisis de los Resultados y Retroalimentación (feedback)*, involucra actividades de análisis y comparación de los resultados y su justificación. Estas actividades se realizan en cada una de las fases principales del procedimiento y termina con una conclusión general después de aplicado al proceso de Gestión de Requisitos y su especificación. Se propone utilizar herramientas y mecanismos de documentación que faciliten la interpretación de los datos y su seguimiento. Finalmente se informa al equipo encargado de la elaboración de los EE y a todo el personal vinculado con este proceso, de los resultados obtenidos con el objetivo de trazar estrategias para el mejoramiento de la GR.

A continuación se describen con mayor profundidad las fases que están el área amarilla del procedimiento RPQ.

2.4 Medición al Proceso

Esta fase está dirigida a la calidad del proceso y tiene un enfoque orientado a metas, por lo que se propone un proceso para medir la calidad en la Gestión de Requisitos que se muestra en la Figura 2.2 utiliza la *Plantilla de calidad – GQM* propuesta y está compuesto básicamente por tres actividades que serán descritas próximamente.

Se debe destacar que las métricas que se proponen en esta fase se calculan a partir de la información que se recoge durante las revisiones que se realizan al proceso. En alguna medida estas métricas también son un indicador relevante de la calidad de los EE que se revisan y por consiguiente de la especificación de requisitos.

Cada una de las actividades que se representan en la Figura 2.2 serán descritas por separadas próximamente, describiendo los procesos, técnicas y herramientas que requieren para su satisfactorio desempeño.

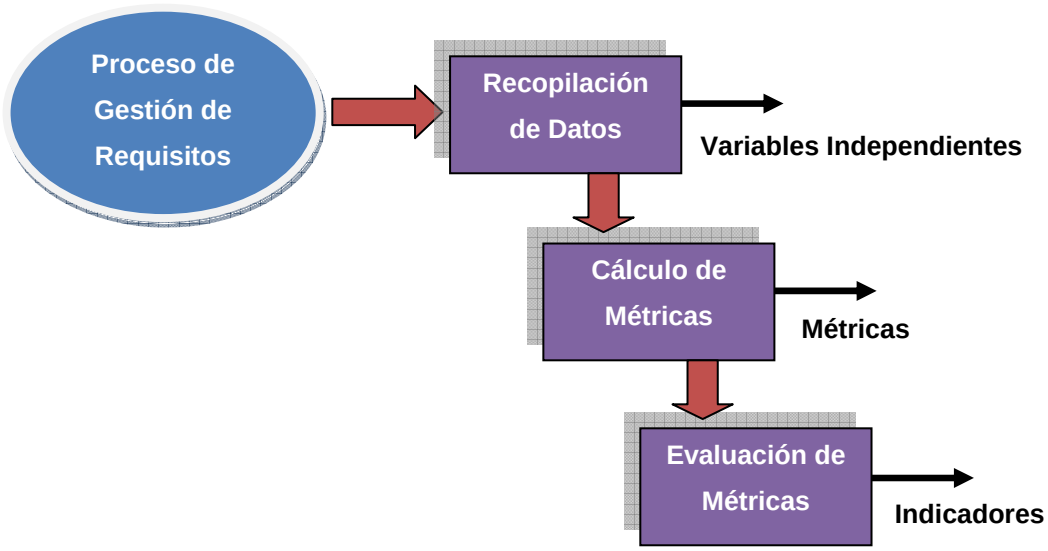

Figura 2. 2 Proceso para medir el proceso de Gestión de Requisitos

2.4.1 Paradigma GQM para la Gestión de Requisitos

Con el Paradigma GQM para la GR, se propone un conjunto de métricas apropiadas que están en función de cada una de las preguntas elaboradas y estas a su vez están asociadas a las metas u objetivos trazados para lograr planificar, controlar, darle un seguimiento y observar si existe mejoramiento o se está avanzando en dirección a la calidad del proceso GR.

Es importante señalar que los datos recolectados de atributos observables (atributos internos y atributos externos, a partir de ahora para esta investigación, se les va a llamar variables independientes (VI) y variables dependientes (VD) respectivamente.

En la siguiente tabla (Tabla 2.1) se muestra la plantilla (Plantilla de Calidad – GQM) que se sugiere usar para representar el enfoque GQM y contiene las metas, preguntas y métricas propuestas para la GR.

Tabla 2. 1 Plantilla de Calidad - GQM

Meta/Objetivo	Pregunta	Métrica
Planificación	¿Cuánto cuesta el proceso de revisión?	EEER, EEES, DPR, DPCD, DPP
	¿Cuánto tiempo consume el proceso de revisión y de GR?	EEER, TPD, EAP
Control y Seguimiento	¿Cuál es la calidad del proceso de revisión y de GR?	DD, EPED, DDP, PDEE, PRNS
	¿En qué medida el personal técnico sigue el procedimiento establecido para las revisiones y la GR?	PRI, PRP, PPR, RPPA, RPPR, PEER, EAP
	¿Cuál es el estado del proceso de revisión?	TEER, PEER
Mejoramiento	¿Cuán efectivo es el proceso de revisión y de GR?	EED, EPED
	¿Cuál es la productividad del proceso de revisión y de GR?	EDE, ER

Mediante el enfoque GQM (y otras herramientas más específicas como la Plantilla de Calidad – GQM), se puede definir y planificar situaciones deseadas, que por medio del análisis de los datos relevantes, permitan evaluar y, en definitiva, mejorar procesos y productos.

2.4.2 Recopilación de Datos

El procedimiento propuesto comienza con la *recopilación de datos* (como se mencionó anteriormente, en esta investigación se adopta el término “variables independientes” para hacer referencia a los datos o medidas). La autora coincide con [Pressman, 2002] [González, 2001] en que este paso se puede considerar el más difícil y se lleva a cabo durante las revisiones planificadas que se realizan a la GR.

Dado que este plan de revisión está inmerso en el Plan de Aseguramiento de la Calidad del Proyecto, la inclusión de métricas al proceso de revisiones (PR) que se van a llevar a cabo durante la GR están encaminadas hacia tres aristas fundamentales, primero para dar seguimiento a la planificación y ejecución de revisiones, segundo para controlar ambos procesos (PR y GR) y tercero para tener una idea amplia de cómo se está avanzado en dirección a la calidad en el proceso donde se está realizando la revisión. Estas actividades propician la toma de acciones correctivas y preventivas para lograr un progreso satisfactorio del proceso.

Se ha diseñado una plantilla (Plantilla de Mediciones) que está compuesta a su vez por tres tablas y cada una de ellas contiene los principales datos que se deben registrar durante las revisiones y que son necesarios para el cálculo de las métricas propuestas. Ver Tabla 2.2., Tabla 2.3 y Tabla 2.4.

Tabla 2.1 Plantilla de Mediciones – Datos para cada revisión

No. Rev.	Fecha	Descripción	Tpo planificado para la revisión	Tpo de duración de la Revisión	Reinspección (Sí / No)	Evaluación

Revisores:	Elemento de Especificación:	Defecto (No conformidades):	Tiempo de corrección

Tabla 2.2 Plantilla de Mediciones – Datos para cada Analista

Nombre Analista	Datos Analista	Semana	Cant. EE planificados

Tabla 2.3 Plantilla de Mediciones – Datos para cada Elemento de Especificación (EE)

Nombre	Descripción	Autor (Analista)

Por otro lado, las VI que se proponen recopilar en la GR se muestran en la Tabla 2.3. En la primera columna se designa el identificador (ID) de la variable, en la segunda el significado y la última columna indica las métricas donde se utiliza.

Tabla 2.4 Variables Independientes

ID	Significado	Referencia
DE	Cantidad de defectos detectados durante una revisión.	Ref. 1a, Ref. 1b, Ref. 10a, Ref. 11a
DT	Cantidad total de defectos encontrados en el proceso.	Ref. 2ª, Ref. 2b
CEE	Cantidad total de elementos de especificación revisados.	Ref. 2a, Ref. 2b
CR	Cantidad de disposiciones de reinspecciones	Ref. 3
CI	Cantidad total de revisiones.	Ref. 3
EER	Cantidad de elementos de especificación revisados en una revisión.	Ref. 4, Ref. 5, Ref. 6a, Ref. 7b, Ref. 9a', Ref. 11a
TDurac	Tiempo de duración de una revisión.	Ref. 6a, Ref. 9a', Ref. 10a, Ref. 12a, Ref. 13, Ref. 14
CERe	Cantidad de elementos de especificación realizados en un sub-sistema.	Ref. 7a, Ref. 9a
Canal	Cantidad de analistas que participaron en un sub-sistema.	Ref. 7a
CRev	Cantidad de revisores que participan en una revisión.	Ref. 7b, Ref. 9a', Ref. 10a
TDurp	Tiempo de duración de un sub-sistema.	Ref. 7a
CEEreal	Cantidad de elementos de especificación realizados hasta el momento.	Ref. 8a
CEEplan	Cantidad de elementos de especificación que el analista debía haber realizado.	Ref. 8a
TEA	Tiempo invertido por un analista en especificar EE.	Ref. 9a
TER	Tiempo invertido por un revisor.	Ref. 9a'
TPrepReal	Tiempo de preparación que invirtió un inspector.	Ref. 10a
TCorrReal	Tiempo real de corrección de defectos.	Ref. 10a
TEstim	Tiempo de duración estimado de una revisión en la planificación.	Ref. 12a, Ref. 14
TCorrec	Tiempo de corrección utilizado por un analista.	Ref. 15
TDE	Total de defectos detectados que pertenecen al analista.	Ref. 15
RNS	Cantidad de revisiones que fueron evaluadas como no satisfactorias en el proceso de GR.	Ref. 17
RP	Cantidad de revisiones que fueron programadas en el Plan de Aseguramiento de Calidad del proyecto, específicamente	Ref. 17

	en el proceso de GR.	

2.4.3 Cálculo de Métricas

Una vez recopiladas las VI se prosigue con el cálculo de las métricas, de acuerdo a la cantidad de VI que se identifiquen será la gama de métricas que se pueden abarcar.

A continuación se detallan las métricas que han sido referenciadas anteriormente para el proceso de GR y sirven para analizar su estado y trabajar en su mejoramiento continuo. En el Anexo 4 se presenta una tabla resumen con las métricas propuestas.

La importancia del conjunto de métricas que se definen, además de que forman parte del modelo de medición que se propone, radica en que la información general que se obtiene de cada una de las métricas, se puede usar para hacer comparaciones de lo planificado con lo real, de la eficiencia de los analistas y revisores que intervienen en el proceso, de la cantidad de defectos que se detecten, entre otros aspectos que se especificarán en cada caso. Estos resultados van a servir para hacer un análisis exhaustivo del proceso y saber hacia dónde enfocar la atención, además para ganar en experiencia.

Las métricas que se presentan han sido tomadas del libro “Metrics and Models in Software Quality Engineering”, [Kan, 2002], adecuándolas a las condiciones del procedimiento RPQ y considerando otros tipos de revisión, pues Kan en su libro únicamente considera las inspecciones. Estas métricas se pueden adaptar y/o aplicar a otros procesos del ciclo de desarrollo de un producto software.

2.4.3.1 Efectividad de Eliminar los Defectos en la Revisión

Se propone en el modelo la métrica ***(1a)*** que da una vista global de la efectividad de la revisión en el proceso que se está analizando, es decir en la Captura de Requisitos (GR).

$$EED_k = \frac{DE_k}{DL} * 100 \quad \textbf{(1a)}$$

$$DL = DE_k + DEP$$

$$DEP = \sum_{k=i+1}^{n} DE_k$$

EED_k : Efectividad de eliminar los defectos en la revisión k.

DE_k : Cantidad de defectos detectados durante la revisión k.

DL : Cantidad total de defectos presentes en el proceso, cuando éste ha sido terminado.

DEP : Cantidad de defectos encontrados posterior a la revisión, es decir la cantidad de defecto encontrados en la n-i restantes revisiones que se indican en el plan de revisiones y auditorías del proyecto. También puede calcularse este valor considerando los defectos detectados en las revisiones efectuadas hasta el momento (k = m, con i < m < n) en que se desea analizar la métrica, pero serán resultados parciales que pueden cambiar al finalizar el proceso.

Si se representan en un gráfico los valores resultantes de la métrica de efectividad se puede analizar qué revisiones de las realizadas al proceso resultan poco efectivas y así valorar la posibilidad de incluirla o no en el mismo flujo de trabajo (FT) de desarrollo o en otros flujos cuyas características sean similares. Además se puede tomar cualquier otra decisión que contribuya al mejoramiento del Proceso de Revisiones.

El comportamiento de esta métrica puede ser representado en un modelo que dé una idea global de la eficiencia de las revisiones. Por ejemplo, se pudiera almacenar los valores de la efectividad de diferentes revisiones realizadas a un proyecto y compararlas entre sí.

2.4.3.2 Efectividad Promedio de Eliminar los Defectos en las Revisiones

Partiendo de la información que ofrece ***(1a)*** se propone determinar la efectividad promedio de las revisiones, considerando un único proceso o múltiples procesos. Esta nueva métrica ***(1b)*** permitirá realizar un análisis sobre el proceso de revisiones y definir acciones que contribuyan al mejoramiento del proceso.

$$EPED = \frac{\sum_{k=1}^{n} EED_k}{n} \quad \textbf{(1b)}$$

$$EED_k = \frac{DE_k}{DL} * 100$$

$$DL = DE_k + DEP$$

$$DEP = \sum_{k=i+1}^{n} DE_k$$

$EPED$: Efectividad promedio de eliminar los defectos en las revisiones, con n (1<=k<=n) que representa la cantidad de revisiones consideradas.

DE_k : Cantidad de defectos detectados durante la revisión k.

EED_k : Efectividad de eliminar los defectos en la Revisión k.

DL : Cantidad total de defectos presentes en el proceso, cuando éste ha sido terminado.

DEP : (Ver en la definición de la métrica anterior)

2.4.3.3 Densidad de Defectos

Se propone la métrica ***(2a)*** que ha sido modificada ubicando en el denominador la Cantidad de Elementos de la Especificación (CEE) pero se puede extender a todos los elementos de configuración (EC) que existen en cada uno de los FT del proyecto en general.

$$DD = \frac{DT}{CEE} \quad \textbf{(2a)}$$

DD : Densidad de defectos detectados.

DT : Cantidad total de defectos encontrados en el proceso.

CEE : Cantidad de elementos de especificación revisados.

Esta métrica da una medida de la proporción de defectos del proceso con respecto a la cantidad de elementos de especificación. Por ejemplo, si en el proceso de captura de requisitos están definidos 5 artefactos (Glosario de Términos, PEN, Especificaciones suplementarias, entre otros) y cada uno de ellos contiene cierta cantidad de elementos de especificación (EE), luego se realiza una revisión al principio de este proceso y se inspecciona uno de estos artefactos detectando varios defectos en cada EE que lo compone, entonces se hace un análisis comparativo entre los diferentes valores de DD obtenidos para definir cuándo los elementos son más denso o menos denso.

De la forma en que esta métrica está definida debe ser evaluada al finalizar el proceso también y puede ser aprovechada como experiencia para procesos y proyectos futuros.

2.4.3.4 Densidad Promedio de Defectos Detectados en el Proceso

Además se propone emplear la métrica ***(2b)*** para tener una idea global de la calidad con que se producen los elementos en el proceso que se desarrolla y de la mejora continua del proceso de revisiones al transcurrir un período de tiempo determinado. Teniendo en cuenta que mientras menor sea el valor de esta métrica, menores son los defectos que son detectados y por consiguiente la calidad de los elementos de especificación revisados es más alta.

$$DDP = \frac{\sum_{k=1}^{n} DD_k}{n} \quad \textbf{(2b)}$$

$$DD_k = \frac{DT_k}{CEE_k}$$

DDP : Densidad de defectos promedio detectados en el proceso de GR, con h (1<=k<=n) que representa la cantidad de elementos considerados.

DD_k : Densidad de defectos en la revisión k.

DT_k : Cantidad total de defectos encontrados en el proceso.

CEE_k : Cantidad de elementos de especificación revisados.

2.4.3.5 Porcentaje de Reinspección (solo para inspecciones)

Se propone la siguiente métrica ***(3)*** que puede ser considerada tanto para análisis del porcentaje de reinspección de un proceso específico, como para el porcentaje de reinspección del proceso en general, considerando todos los proyectos que han sido inspeccionados.

$$PRI = \frac{CR}{CI} * 100 \quad \textbf{(3)}$$

PRI : Porcentaje de reinspección.

CR: Cantidad de disposiciones de reinspecciones, que son indicadas por el equipo de inspección si entiende que son demasiados defectos o muy complejos como para considerar satisfactoria la inspección al proceso (con los elementos correspondientes) que se realiza. En estos casos se evalúa de mal el proceso inspeccionado y se indica realizar nuevamente una revisión cuando hayan sido corregidos los defectos.

CI : Cantidad total de revisiones.

2.4.3.6 Cantidad de EE Revisados

Se propone la métrica ***(4)*** que ayuda en gran medida a controlar la cantidad de elementos de especificación que se revisan en cada una de las revisiones que se realicen en el proceso de GR para realizar análisis en cuanto a las mejoras o deficiencias que se registren, además este valor servirá para realizar posteriores cálculos de diferentes métricas.

$$TEER = \sum_{k=1}^{n} EER_k \quad \textbf{(4)}$$

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k : Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

2.4.3.7 Promedio de EE Revisados

Con la información que se obtiene de esta métrica ***(5)*** se puede realizar un análisis sobre la amplitud de las revisiones, pues da una idea de cuán exhaustivamente se revisó el proceso.

$$PEER = \frac{TEER}{n} \quad \textbf{(5)}$$

$$TEER = \sum_{k=1}^{n} EER_k$$

$PEER$: Promedio de elementos de especificación revisados.

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k : Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

2.4.3.8 Productividad Promedio de la Revisión

Se incluye en el modelo la métrica ***(6a)***, que permite hacer un análisis de la productividad de las revisiones al proceso. Esta expresión ha sido modificada para considerar los elementos de especificación en lugar de las líneas de código.

$$PRP = \frac{TEER}{\sum_{k=1}^{n} TDurac_k} \quad \textbf{(6a)}$$

$$TEER = \sum_{k=1}^{n} EER_k$$

PRP : Productividad promedio de la revisión.

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k : Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

$TDurac_k$: Tiempo de duración de la revisión k.

2.4.3.9 Productividad Promedio en las Revisiones de los Procesos

Además se adiciona la métrica ***(6a')***, para comparar el desempeño de las revisiones realizadas a los diferentes procesos y analizar el comportamiento global de la productividad de las revisiones considerando un grupo de procesos.

$$PPR_{kj} = \frac{\sum_{k=1}^{h} TEER_k}{\sum_{j=i}^{m} \sum_{k=1}^{n} TDurac_{kj}} \quad \textbf{(6a')}$$

$$TEER_k = \sum_{k=1}^{n} EER_k$$

PPR_{kj} : Productividad promedio en la revisión k al proceso j.

$TDurac_{kj}$: Tiempo de duración de la revisión k al proceso j, donde n (1<=k<=n) es la cantidad de revisiones y m (1<=j<=m) representa la cantidad de procesos del proyecto.

$TEER_k$: Cantidad total de elementos de especificación revisados en la revisión k, donde h (1<=k<=h) es la cantidad de procesos.

EER_k : Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

2.4.3.10 Razón de Realización Promedio de los Analistas

Se incluye la métrica ***(7a)*** que se puede aplicar para conocer la razón de realización promedio de todos los analistas que intervienen el proceso de GR, también se puede aplicar de forma independiente a cada uno de los analistas que llevan a cabo un sub-sistema específico del proceso.

$$RPPA = \frac{CEER}{\sum_{k=1}^{n} \frac{TDurp_k}{CAnal_k}} \quad \textbf{(7a)}$$

$$CEER = \sum_{k=1}^{n} CE\,\mathrm{Re}_k$$

$RPPA$: Razón de realización promedio de los analistas.

$TDurp_k$: Tiempo de duración del sub-sistema k.

$CAnal_k$: Cantidad de analistas que participaron en el sub-sistema k.

$CEER$: Cantidad total de elementos de especificación realizados en el proceso.

$CE\,\mathrm{Re}_k$: Cantidad de elementos de especificación realizados en el sub-sistema k. Tal que n representa el número total de sub-sistemas que conforman al proceso de GR (1<=k<=n).

La razón de realización promedio de los analistas en el proceso de GR es un indicador de la cantidad de elementos de especificación (CEE) que realizó un analista en su sub-sistema del total de EE realizado

durante todo el proceso y puede emplearse para la planificación de futuros procesos con características similares al analizado.

2.4.3.11 Razón de Preparación Promedio de los Revisores

Se incluye la métrica ***(7b)*** con objetivos similares a los de la métrica (7ª), aunque para el caso de los revisores o inspectores la característica que marca la diferencia es que se tiene en cuenta la fase de preparación que es la fase de detección de defectos, por lo que esta métrica da un indicador de la cantidad de EE que revisó un revisor del total de EE revisados durante esta fase. Puede emplearse para la planificación de futuras revisiones.

$$RPPR = \frac{TEER}{\sum_{k=1}^{n} \frac{T\,\mathrm{Pr}\,ep_k}{C\,\mathrm{Re}\,v_k}} \quad \textbf{(7b)}$$

$$TEER = \sum_{k=1}^{n} EER_k$$

$RPPR$: Razón de preparación promedio de los revisores.

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k: Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

$T\,\mathrm{Pr}\,epr_k$: Tiempo de preparación de la revisión k.

$C\,\mathrm{Re}\,v_k$: Cantidad de revisores que participan en la revisión k.

2.4.3.12 Eficiencia de los Analistas en el Proceso de GR

Se incluye en el modelo la métrica ***(8)***, que da una idea de si realmente los analistas están aprovechando el tiempo en el cumplimiento de sus actividades y realización de sus artefactos. Al igual que la métrica anterior esta se puede adaptar para saber la eficiencia de los revisores.

$$EAP = \frac{CEEplan}{CEEreal} \quad \textbf{(8)}$$

EAP : Eficiencia de los analistas en el proceso de GR.

$CEEplan$: Cantidad de elementos de especificación que debía haber realizado.

$CEEreal$: Cantidad de elementos de especificación realizados hasta el momento.

Además se puede hacer un análisis sobre el desempeño de los analistas y saber cuáles son los más eficientes.

2.4.3.13 Esfuerzo Promedio de los Analistas por EE

Se propone en el modelo la métrica ***(9a)*** para los analistas, que permite hacer un análisis del esfuerzo invertido en la realización de los elementos de especificación del proceso.

$$EEES = \frac{\sum_{k=1}^{n} TEA_k}{CEER} \quad \textbf{(9a)}$$

$$CEER = \sum_{k=1}^{n} CE\,\mathrm{Re}_k$$

$EEES$: Esfuerzo promedio de los analistas por EE.

TEA_k : Tiempo invertido por el analista k en especificar EE, donde n representa la cantidad de analistas (1<=k<=n). Se refiere a todo el tiempo que lleva el analista hasta el momento especificando EE.

$CEER$: Cantidad total de elementos de especificación realizados en el proceso.

$CE\,\mathrm{Re}_k$: Cantidad de elementos de especificación realizados en el sub-sistema k. Tal que n representa el número total de sub-sistemas que conforman al proceso de GR (1<=k<=n).

2.4.3.14 Esfuerzo Promedio de los Revisores por EE

Se incluye en el modelo la métrica ***(9a')*** para el caso de los revisores, que permite hacer un análisis del esfuerzo invertido en la revisión de los elementos de especificación del proceso.

$$EEER = \frac{\sum_{k=1}^{n} TER_k}{TEER} \quad \textbf{(9a')}$$

$$TEER = \sum_{k=1}^{n} EER_k$$

$EEER$: Esfuerzo promedio por elemento de especificación de los revisores.

TER_k: Tiempo invertido por el revisor k en revisar EE, donde n representa la cantidad de revisores (1<=k<=n). Se refiere a todo el tiempo que lleva el revisor hasta el momento revisando EE.

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k: Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

2.4.3.15 Esfuerzo Promedio por Defectos Detectados

Se propone la utilización de la métrica ***(10)***, que permite analizar el esfuerzo promedio invertido en una revisión en la detección de un defecto por parte de los revisores que intervienen en ella. De igual modo, esta métrica se pudiera emplear para calcular el esfuerzo promedio por defectos detectados en un EE considerando el esfuerzo de realizar un EE y la cantidad de defectos que se detecten en él.

$$EDE = \frac{\sum_{k=1}^{n} Esfr_k}{\sum_{k=1}^{n} DE_k} \quad \textbf{(10)}$$

$$Esfr_k = T\,\mathrm{Pr}epr_k + C\,\mathrm{Re}v_k * TDurac_k + TCorrecr_k$$

$$T\,\mathrm{Pr}epr_k = \frac{\sum_{i=1}^{r} T\,\mathrm{Pr}e\,\mathrm{Re}al_{i,k}}{r}$$

$$TCorrecr_k = \frac{\sum_{j=1}^{a} TCorr\,\mathrm{Re}al_{j,k}}{a}$$

EDE: Esfuerzo promedio por defectos detectados.

DE_k: Defectos detectados en la revisión k.

$Esfr_k$: Esfuerzo de la revisión k, donde n representa la cantidad de revisiones (1<=k<=n).

$C\,\mathrm{Re}\,v_k$: Cantidad de revisores que participan en la revisión k.

$T\,\mathrm{Pr}\,epr_k$: Tiempo promedio de preparación de la revisión k.

$T\,\mathrm{Pr}\,e\,\mathrm{Re}\,al_{i,k}$: Tiempo real que empleó el revisor *i* en la revisión k (en revisar los EE), donde *r* representa la cantidad de revisores (1<=i<=r).

$TCorrecr_k$: Tiempo promedio de corrección de defectos de la revisión k.

$TCorr\,\mathrm{Re}\,al_{j,k}$: Tiempo real de corrección de defectos, donde *a* representa la cantidad de analistas (1<=j<=a).

$TDurac_k$: Tiempo de duración de la revisión k.

2.4.3.16 Promedio de Defectos Detectados por EE

Se incluye en el modelo la métrica ***(11)***, que permite hacer un análisis de los defectos detectados por elementos de especificación revisados.

$$PDEE = \frac{\sum_{k=1}^{n} DE_k}{TEER} \quad \textbf{(11)}$$

$$TEER = \sum_{k=1}^{n} EER_k$$

$PDEE$: Promedio de defectos por elemento de especificación.

DE_k : Defectos detectados en la revisión k.

$TEER$: Cantidad total de elementos de especificación revisados.

EER_k : Cantidad de elementos de especificación revisados en la revisión k, donde n representa el número total de revisiones (1<=k<=n).

El valor obtenido de esta métrica da una idea de la cantidad de defectos encontrados en las revisiones tanto en el proceso que se analiza como en los restantes del proyecto que se desarrolle, por lo tanto sirve para analizar la efectividad del proceso de revisión y además permite analizar el mejoramiento continuo

de la calidad de los procesos, pues los proyectos deben tender a tener cero defectos y en función de esto deben funcionar y contribuir las revisiones.

2.4.3.17 Eficiencia de la Revisión

Se incluye en el modelo la propuesta de la métrica ***(12)*** que permite saber cuán eficientes son las revisiones que se llevan a cabo en el proceso de GR y poder tomar medidas en cuanto al mejoramiento en caso de que se encuentre deficiencias en el proceso de revisiones.

$$ER = \frac{TEstim}{TDurac} \textbf{(12)}$$

ER: Eficiencia de la revisión.

$TEstim$: Tiempo de duración estimado en la planificación de la revisión.

$TDurac_k$: Tiempo real que duró la ejecución de la revisión.

2.4.3.18 Tiempo Promedio de Duración de la Revisión

Se incluye en el modelo propuesto la siguiente métrica ***(13)*** que da una idea del tiempo que se empleó en la realización de revisiones al proceso de GR y esta información será útil entre otras cosas a la planificación de futuras revisiones.

$$TPD = \frac{\sum_{k=1}^{n} TDurac_k}{n} \textbf{(13)}$$

TPD: Tiempo promedio de duración de las revisiones.

$TDurac_k$: Tiempo de duración de la revisión k, donde n representa el número de revisiones (1<=k<=n).

2.4.3.19 Desviación Promedio de la Duración de la Revisión

Se adiciona la métrica ***(14)*** que da una idea de cuán alejada está la planificación de la ejecución real y debe ser considerada para la asignación de tiempo de duración a cada una de las actividades de las revisiones futuras.

$$DPR = \frac{\sum_{k=1}^{n} |TDurac_k - TEstim_k|}{n} \quad \textbf{(14)}$$

DPR : Desviación promedio de duración de la revisión.

$TDurac_k$: Tiempo de duración de la revisión k, donde n representa el número de revisiones (1<=k<=n).

$TEstim_k$: Tiempo de duración estimado de la revisión k en la planificación.

2.4.3.20 Estimación de Tiempo por Corrección de Defectos Detectados en la Especificación de un Analista

Se propone la métrica ***(15)*** que da una idea de cuánto se alejan los analistas de la planificación para corregir defectos detectados en las revisiones. Esta medida es muy importante porque da a conocer el tiempo que puede atrasarse el proceso y por consiguiente los demás procesos del proyecto provocando la entrega tardía del producto software.

$$DPCD_i = \frac{TPCorr_i}{TDE_i} \quad \textbf{(15)}$$

$$TPCorr_i = \frac{\sum_{k=1}^{n-1} TCorrec_{i,k}}{\sum_{k=1}^{n-1} DE_{i,k}}$$

$DPCD_i$: Desviación promedio del analista *i* por corrección de defectos detectados en la especificación.

$TPCorr_i$ **:** Tiempo por corrección de defectos del analista *i*.

$TCorrec_{i,k}$ **:** Tiempo de corrección utilizado por el analista *i* en la revisión *k,* (1<=k<=n-1) donde n-1 es la cantidad de revisiones realizadas sin contar la revisión actual.

$DE_{i,k}$ **:** Cantidad de defectos detectados al analista i en la revisión k, (1<=k<=n-1) donde n-1 es la cantidad de revisiones realizadas sin contar la revisión actual.

TDE_i : Total de defectos detectados que pertenecen al analista *i*.

2.4.3.21 Desviación del Plan del Proyecto

Se propone la métrica ***(16)*** para hallar la desviación del Plan del Proyecto (DPP), seleccionando la *máxima* desviación de los analistas calculada con la métrica anterior ***(15)***. Esto significa que el analista que más se demore en corregir sus defectos es el que va a determinar el buen cumplimiento del Plan provocando atraso en el proceso.

2.4.3.22 Porcentaje de Revisiones No Satisfactorias

Se propone la métrica ***(17)*** que permite conocer la cantidad de revisiones no satisfactorias, analizarlas en detalle para detectar las causas que provocaron la no satisfacción y trazar las acciones correctivas y preventivas correspondientes.

$$PRNS = \frac{RNS}{RP} * 100 \quad \textbf{(17)}$$

$PRNS$: Porcentaje de revisiones no satisfactorias en el proceso de GR.

RNS : Cantidad de revisiones que fueron evaluadas como no satisfactorias en el proceso de GR.

RP : Cantidad de revisiones que fueron programadas en el Plan de Aseguramiento de Calidad del proyecto, específicamente en el proceso de GR.

Las métricas descritas en esta sección son empleadas en el proceso de GR, aunque como se había mencionado anteriormente se pueden extender a todos los procesos del ciclo de desarrollo de un producto software con el objetivo de analizar el estado de las revisiones desde dos perspectivas: a nivel gerencial y a nivel de proyectos específicos. Estas métricas facilitan el análisis de los procesos para trabajar en su mejoramiento continuo.

2.4.4 Evaluación de Métricas

Finalmente, la autora coincide con [Pressman, 2002] al exponer que las métricas se deben evaluar. La evaluación de métricas se centra en las razones de los resultados obtenidos y produce un grupo de

indicadores que guían al proceso. Se hace un análisis comparativo para definir y establecer la escala o ranquin en que se van a encontrar los valores y después ubicar estos valores en el rango que se encuentran. Esto es lo que permite saber si se está bien o mal y dar las conclusiones finales para a partir de esta información tomar las medidas pertinentes.

2.5 Medición al Producto

Esta fase se enfoca a la calidad del producto viendo la especificación como un todo. Para la medición de la calidad del producto se propone un proceso (Figura 2.3) basado en la Metodología de métricas para la calidad del software que prescribe el estándar IEEE 1061[1], para llevar a cabo un proceso de medición de calidad que se ajusta perfectamente a los objetivos de esta investigación. El proceso primeramente tiene en cuenta las características y sub-características de calidad que evalúan a una especificación de requisitos, se identifican las métricas de calidad, luego se implementan y calculan, posteriormente se verifica la validez de las métricas. Cada una de estas actividades se describe de forma independiente más adelante haciendo referencia a los procesos, técnicas y herramientas que se utilizan para su óptimo cumplimiento.

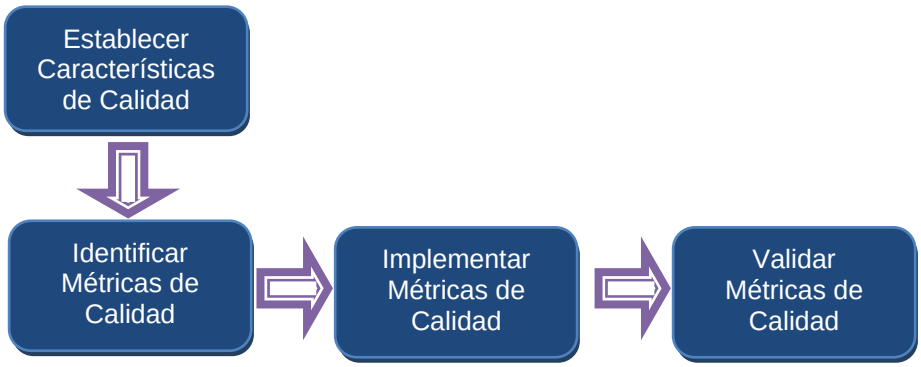

Figura 2. 3 Proceso para medir la calidad de la ERS

El proceso original propone cinco actividades fundamentales [IEEE 1061, 1998]: (1) establecer los requisitos de calidad del software, (2) identificar las métricas para la calidad del software, (3) implementar las métricas de calidad del software, (4) analizar los resultados de las métricas de calidad y finalmente (5)

[1] Esta Metodología de métricas para la calidad de Software prescribe un marco conceptual de descomposición en factores, sub-factores y métricas (Ver Anexo 6).

validar las métricas de calidad del software. En esta propuesta se hace una adaptación donde la cuarta actividad se elimina de este proceso porque se estableció en el procedimiento general que fue descrito en el epígrafe 2.3 como una de sus fases principales que se lleva a cabo durante todo el procedimiento, es decir, que se aplica a las fases de medición al proceso y medición al producto respectivamente. De igual manera esta actividad se une a la Retroalimentación que permite informar a todo el personal involucrado acerca de los resultados y cambios o correcciones que se deben hacer.

En esta fase se puede emplear un procedimiento para el cálculo de un "factor de calidad" [Buglione & Abran, 1999] que integra tres dimensiones (Usuarios, Desarrolladores y Administradores) y se realiza a través del completamiento de cuestionarios (Ver Anexo 5).

2.5.1 Establecer Características de Calidad

La primera actividad que aparece en la figura (Figura 2.3) es *Establecer Características de Calidad* para la especificación. Esto significa identificar y determinar la lista de las posibles características de calidad que se deseen evaluar en la especificación.

Se hizo un análisis exhaustivo de las características de calidad que se pueden medir y como resultado se conjugan las características que propone el marco ISO/IEC 9126 con las del estándar IEEE 830 cuya relación se muestra en la Tabla 2.8 y refleja que la especificación se mide por el grado en que presenta cierto nivel de correctitud, mantenibilidad, completitud, funcionalidad, entre otras y que entre ellas existe una estrecha relación, por lo que se pueden sacar deducciones a partir de un valor determinado de cierta característica que se relaciona con otra indistintamente.

2.5.1.1 Marco ISO/IEC 9126 (conjunto al Estándar IEEE 830) para la ERS

A continuación se presentan dos tablas, la primera tabla (Tabla 2.6) contiene las definiciones de las características y sub-características de calidad del estándar ISO/IEC 9126 que se aplican y se pueden medir en la ERS, la segunda tabla (Tabla 2.7) contiene básicamente la misma información pero con una descripción sintética conforme al mismo estándar. A partir de estas tablas se determinan las relaciones que existen entre cada una de las sub-características definidas con las ocho características del estándar IEEE 830 (Ver Tabla 2.8).

Tabla 2.5 Definiciones de características y sub-características del estándar ISO/IEC 9126 propuestas para la ERS

1. **Funcionalidad de la Especificación**	
	Capacidad de la ERS para reflejar las funciones que satisfacen las necesidades declaradas e implícitas bajo las condiciones especificadas. Es el grado en que la ERS satisface las necesidades indicadas por las siguientes sub-características:
a. ***Idoneidad***	Capacidad de la ERS para contener un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados. Se relaciona con las siguientes características de la IEEE 830: *correcta*, *no-ambigua* y *completa*.
b. ***Exactitud***	Capacidad de la ERS para reflejar efectos o resultados correctos o convenidos con el grado de exactitud necesario. Se relaciona con la siguiente característica de la IEEE 830: *correcta*.
c. ***Interoperabilidad***	Capacidad de la ERS para interactuar recíprocamente con una o más especificaciones. Se relaciona con la siguiente característica de la IEEE 830: *consistente*.
d. ***Seguridad (informática)***	Capacidad de la ERS para proteger información que contiene cuando la misma sea sensible o limitada, para que personas o sistemas desautorizados no puedan leer o pueden modificarla, y las personas o sistemas autorizados tenga el acceso a ellos. Se relaciona con la siguiente característica de la IEEE 830: *completa*.
2. **Confiabilidad de la Especificación**	
	Capacidad de la ERS para mantener un nivel de realización especificado cuando se usa bajo las condiciones especificadas. Cantidad de tiempo que la ERS está disponible para su uso. Se refiere a las sub-características:
a. ***Madurez***	Capacidad de la ERS de evitar un rechazo total de la misma como resultado de haberse detectado un conjunto de defectos en partes de la especificación. Se relaciona con las siguientes características de la IEEE 830: *organizada* y *verificable*.
3. ***Usabilidad de la Especificación***	
	Grado con que la ERS es fácil de usar. Viene reflejado por las siguientes sub-características:

a. ***Comprensibilidad***	Capacidad de la ERS para permitirle al usuario entender si la especificación es idónea y fácil de comprender. Se relaciona con las siguientes características de la IEEE 830: *no-ambigua*, *completa* y *modificable*.
b. ***Cognoscibilidad***	Capacidad de la ERS para permitirle a usuarios y desarrolladores aprender a usarla. Se relaciona con las siguientes características de la IEEE 830: *no-ambigua*, *completa* y *modificable*.
4. ***Mantenibilidad de la Especificación***	
	Capacidad de la ERS de ser modificada. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a cambios en el ambiente, así como en los requisitos y especificaciones funcionales. Está reflejada por las siguientes sub-características:
a. ***Analizabilidad***	Capacidad de la ERS de ser objeto de un diagnóstico para detectar deficiencias o defectos en la especificación, o para identificar las partes o elementos que van a ser modificados. Se relaciona con las siguientes características de la IEEE 830: *no-ambigua, organizada, modificable* y *trazable*.
b. ***Cambiabilidad***	Capacidad de la ERS para permitir la aplicación de una modificación especificada en los requisitos u otros elementos de especificación. Se relaciona con las siguientes características de la IEEE 830: *organizada* y *modificable*.
c. ***Estabilidad***	Capacidad de la ERS para minimizar los efectos inesperados de las modificaciones o cambios realizados en sus elementos. Se relaciona con la siguiente característica de la IEEE 830: *organizada.*
d. ***Contrastabilidad***	Capacidad de la ERS para permitir la validación de requisitos u otros elementos de especificación. Se relaciona con la siguiente característica de la IEEE 830: *correcta, verificable, modificable*.

Conformidad	Capacidad de la ERS para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares relativas a la funcionalidad, confiabilidad, usabilidad y mantenibilidad respectivamente. Se relaciona con la siguiente característica de la IEEE 830: *completa*.

La Conformidad es una sub-característica que está presente en todas las características que se mencionaron anteriormente, por eso se representa independiente al final de la tabla como factor común, igual ocurre con la Tabla 2.7.

Tabla 2.6 Características y Sub-características de la ISO/IEC 9126 basado en preguntas para la ERS

Característica	Pregunta Central	Sub-característica	Pregunta Central
Funcionalidad	¿Las funciones y propiedades satisfacen las necesidades explícitas e implícitas?	Idoneidad	¿Tiene el conjunto de funciones apropiadas para las tareas especificadas?
		Exactitud	¿Hace lo que fue acordado en forma esperada y correcta?
		Interoperabilidad	¿Interactúa con otras ERS (de partes y componentes) especificadas?
		Seguridad	¿Previene accesos no autorizados a la información que contiene?
Confiabilidad	¿Puede mantener el nivel de rendimiento, bajo ciertas condiciones y por cierto tiempo?	Madurez	¿Tiene un número de defectos tal que permite aún la interpretación correcta de la misma?
Usabilidad	¿Es fácil de usar y de aprender?	Comprensibilidad	¿Es fácil de entender y reconocer la estructura, lógica y su aplicabilidad?
		Aprendibilidad	¿Es fácil de aprender a usar?
Mantenibilidad	¿Es fácil de modificar y revisar?	Analizabilidad	¿Es fácil diagnosticar un error o identificar elementos a modificar?
		Cambiabilidad	¿Es fácil de modificar y adaptar?
		Contrastabilidad	¿Es fácil de revisar y son fáciles de validar las modificaciones?
		Estabilidad	¿Hay riesgos o efectos inesperados cuando se realizan cambios?

Conformidad	¿Está de acuerdo con las leyes o normas y estándares, u otras prescripciones?

En la tabla que se presenta seguidamente (Tabla 2.8), se representa la fortaleza de la relación que existe entre las características y sub-características definidas para la especificación con las que propone el estándar IEEE 830 (correctitud, no-ambigüedad, completitud, consistencia, importancia/estabilidad, verificabilidad, modificabilidad, trazabilidad). Los círculos en negro indican una relación fuerte, mientras que los grises significan una relación media y los blancos reflejan una relación débil. Esta clasificación se realizó con el objetivo de saber cuánto influyen unas características sobre otras y poder sacar conclusiones. Esto significa que si se han obtenido niveles de calidad bajo, medio y alto (en el peor de los casos) para las características correctitud, no - ambigüedad y completitud respectivamente del estándar IEEE 830, se puede afirmar que la ERS cumple con la SC idoneidad del estándar ISO 9126, es decir, esta SC está presente en la especificación. De es misma forma funciona para las restantes C y SC.

Tabla 2.7 Relación entre las Características ISO/IEC 9126 y las Características de la IEEE 830

Características IEEE 830 / Características ISO/IEC 9126	Correcta	No-ambigua	Completa	Consistente	Ordenada por Importancia/Estabilidad	Verificable	Modificable	Trazable
Funcionalidad								
Idoneidad	○	●	●					
Exactitud	●							
Interoperabilidad				●				
Seguridad			●					
Conformidad con la funcionalidad			●				●	
Fiabilidad o Confiabilidad								
Madurez					●	○		●
Tolerancia al defecto	○		●	●				
Conformidad con la confiabilidad			●					
Usabilidad								
Comprensibilidad		●	●				●	

Aprendibilidad		●	●				●	
Conformidad con la Usabilidad			●					
Mantenibilidad								
Analizabilidad		●			○		●	●
Cambiabilidad					●		●	
Facilidad de revisión y verificación	●					●	●	
Estabilidad					●			
Conformidad con la mantenibilidad			●					

2.5.2 Identificar las Métricas de Calidad

Luego se pasa a la segunda actividad – *Identificar las métricas de calidad* – donde se tiene en cuenta la aplicación de un marco de trabajo adecuado, se debe realizar un análisis de costo-beneficio, es decir, se identifican los costos de implementar las métricas así como los beneficios de aplicar las métricas y por último se ajusta el conjunto de métricas que se van a emplear.

Las métricas de calidad que se identificaron son las que se muestran a continuación [de Antonio, 2000].

2.5.2.1 Confiabilidad

$$Confiabilidad = 1 - \frac{DL}{CEER}$$

DL : Cantidad de defectos detectados

$CEER$: Cantidad de elementos de especificación realizados.

Otra medida que se puede considerar para esta característica es:

- ♠ Cantidad de defectos detectados en la documentación

2.5.2.2 Mantenibilidad

$$Mantenibilidad = 1 - 0.1 * PDHCorr$$

$$PDHCorr = \frac{\sum_{i=1}^{n} TPCorr_i}{n}$$

$$TPCorr_i = \frac{\sum_{k=1}^{n-1} TCorrec_{i,k}}{\sum_{k=1}^{n-1} DE_{i,k}}$$

$PDHCorr$: Promedio días-hombre por corrección

$TPCorr_i$: Tiempo por corrección de defectos del analista *i*.

$TCorrec_{i,k}$: Tiempo de corrección utilizado por el analista *i* en la revisión *k,* (1<=k<=n-1) donde n-1 es la cantidad de revisiones realizadas sin contar la revisión actual.

$DE_{i,k}$: Cantidad de defectos detectados al analista i en la revisión k, (1<=k<=n-1) donde n-1 es la cantidad de revisiones realizadas sin contar la revisión actual.

Otras medidas que se pueden considerar para esta característica son:

- ♠ Tiempo medio de cambio
- ♠ Cantidad de defectos detectados sin resolver
- ♠ Porcentaje de cambios que introducen defectos
- ♠ Cantidad de módulos afectados por el cambio

2.5.2.3 Flexibilidad o Modificabilidad

$Modificabilidad = 1 - 0.05 * PDHCamb$

$PDHCamb$: Promedio días-hombre por cambio

2.5.2.4 Nivel de Calidad de las Características del Estándar IEEE 830

Por otro lado, para conocer el grado o nivel de calidad de las características del estándar IEEE 830 se recomienda la siguiente métrica:

$$NC = \frac{\sum_{i=1}^{n} SI}{n}$$

Esto significa que el nivel de calidad de una característica se obtiene dividiendo la sumatoria de respuestas SI de su lista de chequeo por la cantidad total de preguntas para la misma característica.

2.5.2.5 Nivel de Calidad Global de la ERS

Los niveles de calidad para cada característica que se obtuvieron anteriormente se pueden agrupar convenientemente para producir al final el resultado global. El nivel de calidad global de la especificación (NCE) representa el grado de satisfacción de todas las características explícitas e implícitas y se calcula a través de la siguiente expresión:

$$NCE = \frac{\sum_{i=1}^{8} NC_i}{8}$$

Para esta métrica se suman los valores obtenidos de la métrica anterior para cada una de las características y se divide por ocho, que es el total de características que propone el estándar.

Es importante resaltar que estas métricas son subjetivas porque evaluadas por distintas personas pueden obtenerse resultados diferentes, sin embargo son una vía simple y significativa para evaluar la calidad de la especificación.

2.5.3 Implementar las Métricas de Calidad

En la tercera actividad – *Implementar las métricas de calidad* – se define el procedimiento de recolección de datos, para este caso, como herramienta de recopilación de datos para algunas métricas y técnica de elicitación se utilizan las plantillas que contienen los datos que fueron obtenidos en la fase anterior y para las restantes métricas, se utiliza es la lista de chequeo (ver Anexo 7) que se diseñó para la ERS respectivamente; una vez que se haya definido el medio para recolectar los datos se prosigue con la recolección y cálculo de los valores de las métricas.

2.5.4 Validar las Métricas de Calidad

Esta actividad es sumamente importante pero además es compleja. En cuestión, se deben aplicar una metodología y criterios de validez, además de un procedimiento de validación, donde se identifican la muestra de factores de calidad, la muestra de métricas, se realiza un análisis estadístico, luego se documentan los resultados, se pasa a la revalidación de las métricas para terminar con una evaluación de la estabilidad del entorno.

Básicamente, existen dos estrategias para corroborar o falsear la validez de las métricas [Olsina, 1999]: la teórica y la empírica. A su vez, dos enfoques comúnmente empleados para validación se corresponden con los atributos internos de los entes, en donde se dice que una medida con fines de evaluación es válida internamente, o “válida en un sentido estrecho”, o, por otra parte, con las características externas de más alto nivel (por ejemplo, costo, calidad), en donde se dice que una medida es válida externamente, o “válida en un sentido amplio”. En esta última categoría entran las métricas empleadas en los modelos predictivos.

La validación teórica permite confirmar que la medida no viola las propiedades de los sistemas relacionales empíricos y numéricos y los modelos de definición de la medición introducidos anteriormente.

La validación empírica permite corroborar una medida por medio de la observación y la planificación de experimentos, para ver por ejemplo, si los desarrolladores o usuarios concuerdan con la existencia de algún atributo, o si el mapeo del mundo real al modelo mental empírico es una representación adecuada del atributo, o si el tipo de escala es el adecuado en el contexto del proyecto de evaluación (entre otros aspectos, como la determinación de si una métrica de un atributo interno puede ser usada para predecir el valor de una variable dependiente -característica externa) [Olsina, 1999].

Por otro lado, la autora comparte el mismo criterio de Kitchenham [Olsina, 1999] asumiendo que para que una medida sea válida se deben mantener estas dos condiciones: 1) que la medida no viole cualquier propiedad necesaria de sus elementos; 2) que cada modelo usado en el proceso de medición debe ser válido.

2.6 Análisis de los Resultados y Retroalimentación

En esta fase se realizan actividades de análisis y comparación de los valores obtenidos después de calculadas las métricas propuestas. Se justifican los resultados alcanzados y a partir de las metas establecidas, el proceso culmina con las conclusiones y recomendaciones del caso de estudio.

Se interpretan los resultados de las fases de medición al proceso y medición al producto de forma independiente y luego de forma general, identificando la calidad de la GR y su especificación; al mismo tiempo se hacen predicciones de la calidad, eficiencia, productividad, entre otros, asegurando la conformidad con las metas y características de calidad establecidas.

Todas estas métricas que se identificaron deben dar como resultado un número dentro del intervalo [0 - 1], indicando el cero la ausencia de la característica y el uno la presencia absoluta de la característica. Se establece la siguiente clasificación (Tabla 2.9) para ubicar los valores que se obtienen después de calculada cada métrica propuesta. Cuando un valor se encuentre en el primer rango, significa que esa característica tiene un alto nivel de presencia en la especificación, mientras que si el valor se encuentra en el segundo rango la característica está en un nivel medio y se considera que también está presente en la especificación, por otro lado si el valor cae en el tercer rango, la característica tiene un bajo nivel y se considera que está completamente ausente de la especificación.

Tabla 2.8 Rango establecido para las características de calidad

Rango		Clasificación	Clasificación Global
1	[0.61 – 1]	Alto	Presente
2	[0.41 - 0.6]	Medio	
3	[0 - 0.4]	Bajo	Ausente

El próximo capítulo está dedicado a esta fase considerando la fase de medición al proceso, es decir, se realiza un análisis de los resultados obtenidos después de haberse revisado los Procesos Elementales del Negocio (PEN) del proyecto CICPC que se lleva a cabo en la Facultad 8 y recoger toda la información que se pudo para el cálculo de las métricas.

Por otro lado, en esta fase también se tiene en cuenta la actividad de retroalimentación donde se informa a todos los miembros encargados de la elaboración de los EE (analistas) acerca de los resultados, las anomalías y se debe trazar un plan para la corrección de errores en caso de estar presentes y planificar los cursos de capacitación necesarios para los analistas que presentaron problemas en sus EE, entre otras medidas que se decidan emprender.

2.7 Conclusiones del Capítulo

En este capítulo se presentó el procedimiento ***RPQ*** y se describieron las fases que lo componen. Además se abordaron los procesos de medición para la Gestión de Requisitos y la ERS, detallando cada uno de los modelos, procesos, herramientas y técnicas que lo conforman. Se considera que las métricas propuestas para evaluar el proceso de GR cubren la mayor parte de sus actividades. Por un lado, las revisiones son una vía práctica y confiable para llevar a cabo el proceso de medición, permitiendo recoger los datos que se necesitan almacenar a través de las plantillas de mediciones diseñadas con este fin y lo más importante, detectando todos los defectos o no-conformidades presentes en la GR. Por otra parte, las métricas de calidad expuestas para evaluar la ERS como producto utilizan principalmente medidas que se recolectaron en la fase anterior por lo que existe una estrecha relación entre ambas fases. De igual forma, las métricas propuestas para calcular el grado o nivel de calidad de cada una de las características del estándar IEEE 830 sienta sus bases en el completamiento de la lista de chequeo confeccionada para lograr este objetivo y a pesar de que son métricas subjetivas, brindan un gran ayuda al analista acerca de la calidad con la que ha confeccionado sus elementos de especificación. Otro aspecto subyacente es que se estableció una relación entre las características de los estándares ISO/IEC 9126 e IEEE 830, por lo que conociendo de la calidad de una característica se puede conocer de la(s) que se relacionan con ella. Se termina con el análisis e información de los resultados obtenidos para la toma de acciones en función de lograr las metas que se propusieron alcanzar.

Capítulo 3. Resultados obtenidos después de aplicar la fase de medición al proceso del procedimiento RPQ en un Proyecto

3.1 Introducción

En este capítulo se hace una valoración de los resultados obtenidos después de aplicar parte del procedimiento propuesto, específicamente el proceso para medir la calidad en la GR que está contenido en la fase de Medición al proceso, ya que el proyecto objeto de estudio (CICPC[1]) todavía se encuentra en los inicios del proceso de Captura de Requisitos. Por parte del personal de calidad del proyecto se planificó una revisión y se conformó un equipo de revisores que se encargaron de realizar la revisión a uno de los artefactos que se producen durante este proceso. Los resultados se ilustran a través de gráficos representativos, analizando las causas que originaron los defectos o no-conformidades detectadas y trazando una línea para corregir estas deficiencias, tanto en los EE como en los analistas encargados de su construcción. Además se evalúa el proceso de revisión efectuado.

3.2 Artefacto objeto de estudio

Durante la Captura de Requisitos se realizan diferentes artefactos que complementan la ERS, entre ellos se encuentran: el glosario de término, especificaciones suplementarias, el plan de desarrollo y un documento que recoge los Procesos Elementales del Negocio (PEN). La fase de medición al proceso se le aplicó a este artefacto. En el documento revisado se describen 27 procesos, de los cuales cuatro han sido denominados procesos claves y los 23 restantes son procesos de apoyo. Ambos tipos de procesos se desglosan a su vez en un conjunto de sub-procesos, cada sub-proceso es especificado en varios Casos de Usos del Negocio (CUN). En total, el PEN contiene 204 CUN que son los elementos de especificación (EE) que se van a tomar como muestra para realizar todo el proceso de medición. Estos EE fueron realizados por 10 analistas, en un período de tiempo de dos meses aproximadamente, trabajando 12 hrs. diarias mínimo, bajo la supervisión del analista principal y el líder del proyecto respectivamente.

[1] El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) es una institución policial venezolana cuyo principal objetivo es garantizar la eficiencia en la investigación del delito, asegurando que el ejercicio de la acción penal conduzca a una sana administración de justicia.

3.3 Aplicación de la Fase de Medición al Proceso

En esta sección se aplican las actividades que conforman al proceso de medición propuesto para evaluar al proceso de Gestión de Requisitos y que fueron descritas en el epígrafe 2.4.

3.3.1 Recopilación de Datos

Para iniciar el proceso de medición en la Gestión de requisitos se planificó una revisión al artefacto objeto de estudio en la que participaron 18 revisores que detectaron los defectos o no-conformidades presentes en los EE (CUN), empleando para ello una lista de chequeo (Ver Anexo 8) y una plantilla de no-conformidades (Ver Anexo 9) para registrar los problemas presentes. Con esta información se completaron las plantillas de medición diseñadas (Ver Anexo 10, Anexo 11 y Anexo 12) que permiten recopilar todos los datos necesarios para el posterior cálculo de las métricas. Algunas variables independientes se muestran el la Tabla 3.1.

Tabla 3.1 Variables Independientes

Variables Independientes		
CRev	Cantidad total de revisiones.	18
TDurac	Tiempo de duración de la revisión.	28hrs
CERe	Cantidad de elementos de especificación realizados en el proceso.	204
CAnal	Cantidad de analistas que participaron en la realización de los EE.	10
TEstim	Tiempo de duración estimado de una revisión en la planificación.	40hrs

3.3.2 Cálculo de las Métricas

A continuación se muestra la Tabla 3.2 que recoge las métricas que fueron calculadas teniendo en cuenta que solo se realizó una revisión, esto implica que muchas de las métricas propuestas en el proceso de medición para la Gestión de Requisitos no se pudieron aplicar.

Tabla 3.2 Valores obtenidos de las métricas

Métrica	Nombre	Valor	Ref.	Observaciones
TEER	Cantidad de elementos de especificación revisados.	204	Ref. 4	Se tienen en cuenta la cantidad de EE que presenta cada proceso del PEN.
DL	Cantidad total de defectos detectados durante la revisión.	574	Ref. 1b	Se tienen en cuenta la cantidad de defectos detectados en cada proceso del PEN.
DD_{EE}	Densidad de Defecto.	2.81	Ref. 2a	Se tiene en cuenta TEER y DL.
CER_e	Cantidad de EE realizados por analista.	Ver Gráfico	Ref. 7a	Se tiene en cuenta los EE que realizó cada analista en cada uno de los procesos donde participó.
$PEER_A$	Promedio de EE realizados por analista.	20.4	Ref. 5	Se tiene en cuenta la cantidad total de EE realizados y la cantidad de analistas.
DT_A	Cantidad de defectos que introdujeron los analistas.	Ver Gráfico	Ref. 2a	
PDEE	Promedio de defectos por analista.	56.9	Ref. 11a	Se tiene en cuenta DT_A y la cantidad de analistas.
DD_A	Densidad de defectos por analista.	Ver Gráfico	Ref. 2a	Se tiene en cuenta DT_A y CER_e
DDP_A	Densidad promedio por analista.	2.73		Considerando DD_A y la cantidad de analistas.
CER_R	Cantidad de EE revisados por revisor.	Ver Gráfico	Ref. 7a	Se considera la cantidad de EE que revisó cada revisor.
$PEER_R$	Promedio de EE revisados por revisor.	11	Ref. 5	Se tiene en cuenta CER_R y la cantidad de revisores.
DL_R	Cantidad de defectos detectados por revisor.	Ver Gráfico	Ref. 1a	Se considera la cantidad de defectos que detectó cada uno de los revisores en los EE que revisó.

$PEER_{DD}$	Promedio de defectos detectados por revisor.	28.55	Ref. 5	Se tiene en cuenta DL_R y la cantidad de revisores que intervienen en el proceso de revisión.
DD_R	Densidad de defecto por revisor.	Ver Gráfico	Ref. 2a	Se tiene en cuenta CER_R y $PEER_{DD.}$
DDP_R	Densidad promedio por revisor.	3.04	Ref. 2b	Considerando DD_R y la cantidad de revisores.
PRP	Productividad promedio de la revisión.	7.29	Ref. 6a	Se considera TEER y el tiempo de duración de la revisión.
RPPA	Razón de realización promedio de los analistas.	5.1	Ref. 7a	Se tiene en cuenta TEER, el tiempo de duración en realizar los EE (720 hrs., considerando que se realizaron en dos meses y que los analistas trabajaban 12 hrs. diarias) y la cantidad de revisores.
EEER	Esfuerzo promedio de los revisores por EE.	0.52	Ref. 9a'	Se tiene en cuenta el tiempo que empleó cada revisor en revisar sus EE y TEER.
ER	Eficiencia de la revisión.	1.43	Ref. 12a	Se tiene en cuenta el tiempo de duración de la revisión y el tiempo planificado de la revisión.

Para el cálculo de la métrica EEER se necesitan los datos que se muestran en la siguiente tabla (Tabla 3.3).

Tabla 3.3 Tiempo empleado por los revisores en revisar sus EE

No	Revisor	TER (hrs)	No	Revisor	TER (hrs)
1	Liudmila	12	10	Maylen	4
2	Ángel Alberto	4	11	Haydée	4
3	Yinet	12	12	Abel Méndez	4

4	Liana	12	13	Erick Eduardo	4
5	Carlos Manuel	4	14	Yendi	4
6	Yainerys	8	15	Greilan	4
7	Yudenia	12	16	Jorge Amado	4
8	Pedro	6	17	Dairy	4
9	Pablo	6	18	Leonardo	2
Sub-total		76	**Sub-total**		30
Total		**106**			

3.3.3 Evaluación de las Métricas

Esta actividad en algunos casos no se realizó porque la información fue insuficiente para establecer los indicadores y escalas. Como solo se llevó a cabo una única revisión y no hay valores antecedentes que permitan el análisis comparativo requerido, no se pueden definir los rangos en los que se pueden ubicar los valores obtenidos. Se determinaron los rangos para la clasificación de las métricas: Densidad de Defectos por EE (**DD_{EE}**), Densidad de Defectos por Analista (**DD_{A}**) y Densidad de Defectos por Revisor (**DD_{R}**) los cuales se muestran a continuación.

Rango para DD_{EE}

	Rango	**Clasificación**
1	[3.33 - 5]	Alto
2	[1.67 - 3.32]	Medio
3	[0 -1.66]	Bajo

Rango para DD_{A}

	Rango	**Clasificación**
1	[4.67 - 7]	Alto
2	[2.34 - 4.66]	Medio
3	[0 - 2.33]	Bajo

Rango para DD_{R}

	Rango	**Clasificación**
1	[10.01 - 14]	Alto
2	[4.01 - 10]	Medio
3	[0 - 4]	Bajo

3.4 Análisis de los Resultados

Los gráficos que se muestran a continuación representan los resultados obtenidos después del cálculo de las métricas empleadas. Se analizan los resultados por gráfico, exponiendo brevemente un resumen en cada caso.

3.4.1 Densidad de Defectos por EE

Este gráfico (Figura 3.1) da una vista general de la distribución de defectos o no-conformidades por los EE realizados. Se observa claramente que la mayoría de los EE están en el rango de defectos entre [0 - 5], por lo que de forma general no presentan muchos defectos. La densidad de defecto es de 2.81 por lo que se ubica en el segundo rango, indicando una densidad media, entonces se puede concluir que de forma general los EE no presentan una mala calidad.

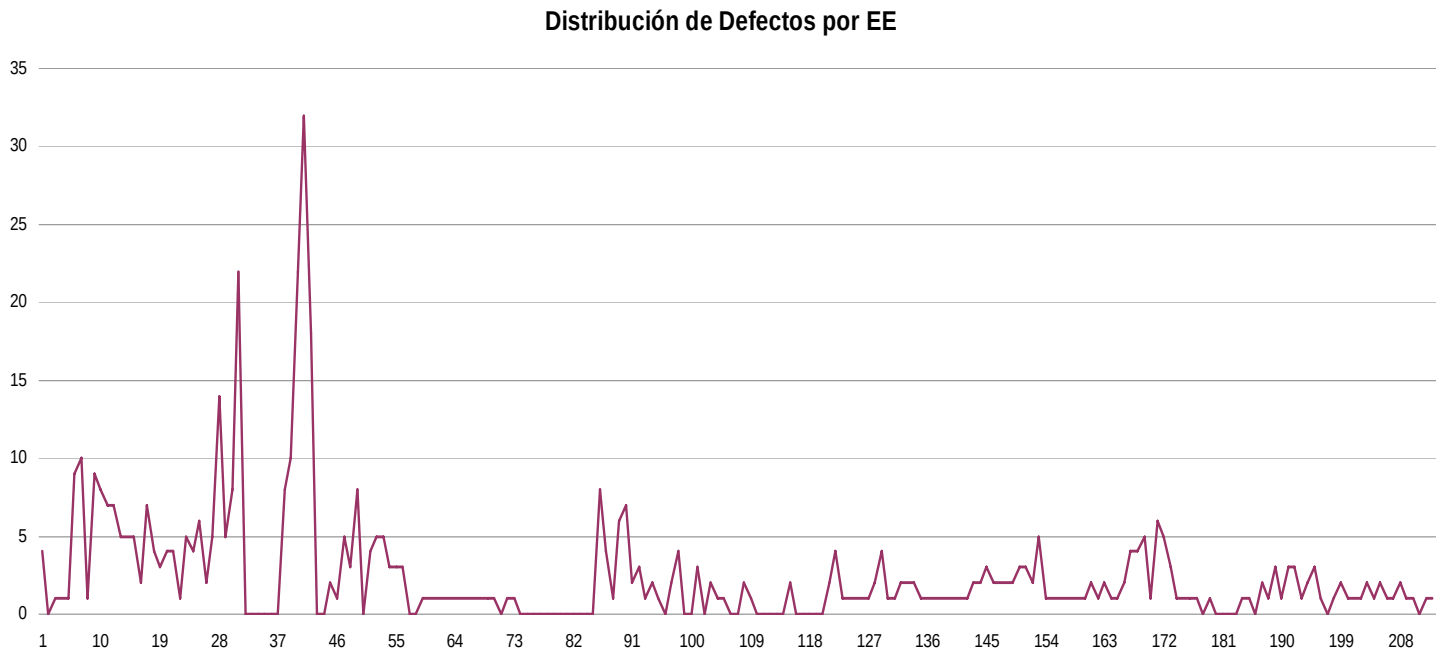

Figura 3. 1 Distribución de los defectos por EE

3.4.2 Cantidad de defectos detectados por procesos clave

Este gráfico (Figura 3.2) muestra la cantidad de defectos que se detectaron en los procesos clave del PEN. Se puede concluir que el proceso clave que más afectado se encuentra por la cantidad de errores o no conformidades presentes es el PC-3 (Investigaciones Criminalísticas), con 194 no conformidades en sus 32 CUN, representando el 34.09% del total de errores encontrados en todos los EE. Se debe resaltar que es el proceso que más EE contiene y que fue llevado a cabo por los analistas Yunexis y Miguel A. En un segundo lugar de criticidad se encuentra el PC-2 (Investigaciones Penal e Interna) con un total de 91 no conformidades, representando un 15.99% del total. Este proceso contiene 18 CUN y lo especificaron los analistas Yenisleydis y Diana.

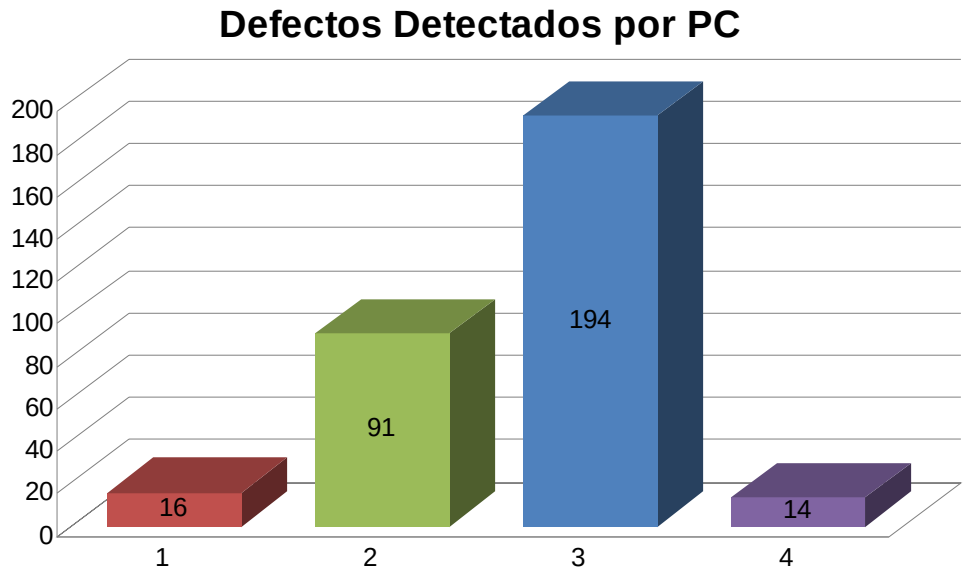

Figura 3. 2 Cantidad de defectos detectados por procesos clave

3.4.3 Cantidad de EE por Analista

Este gráfico (Figura 3.3) representa la cantidad de EE que realizó cada uno de los analistas que participó en el proceso de Captura de Requisitos. El analista que más EE realizó fue Reynaldo con 37 EE de un total de 204, representando un 17.37%. Muy próximo a él, estuvo la analista Arelys con un total de EE

realizados de 31, representando un 14,55%. Los analistas que más se destacaron después de los mencionados anteriormente fueron Miguel A y Yeni con 29 EE y 23 EE respectivamente. Mientras que los analistas que menos EE realizaron fueron Yainelis con 9 EE y Diana con 10 EE. El promedio de EE por analista fue de 20.4 EE. En la Tabla 3.4 se muestra el orden en que se representan los analistas en los gráficos que se muestran en este epígrafe.

Tabla 3.4 Orden de aparición de los analistas en los gráficos

No	Analista	No	Analista
1	Yainelis	6	Miguel A
2	Yadiel	7	Rey
3	Yeni	8	Susel
4	Diana	9	Lizandra
5	Yunexis	10	Arelys

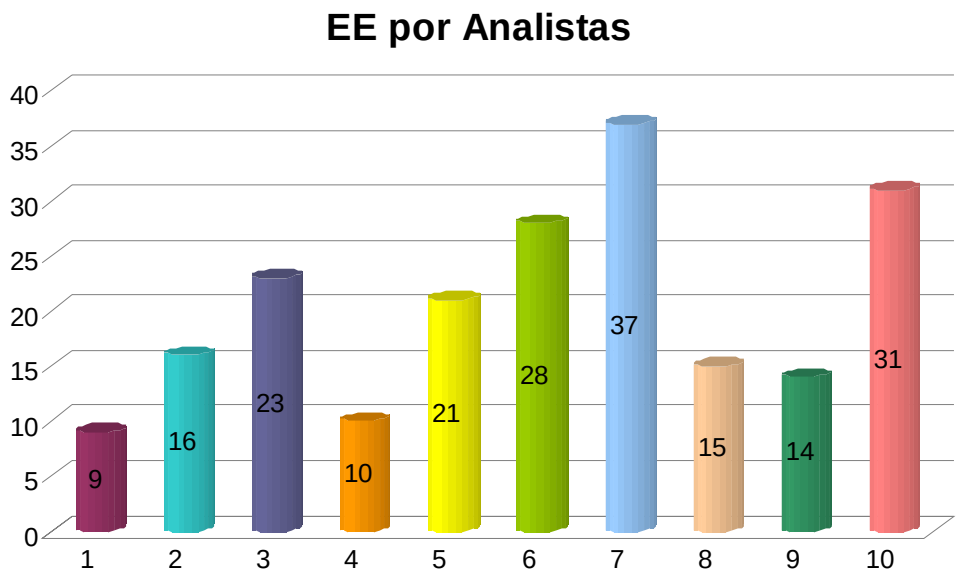

Figura 3. 3 Cantidad de EE por analista

3.4.4 Densidad de Defectos por Analista

Este gráfico (Figura 3.4) muestra la proporción de defectos teniendo en cuenta la cantidad de EE realizados por los analistas y la cantidad de no-conformidades que se le detectaron a cada analista. Como se observa en el gráfico, el analista que presenta más densidad de defectos en sus EE con 6.71 es Miguel A, considerando que realizó 28 EE en los que se le detectaron 188 no-conformidades, esto lo ubica en el primer rango establecido para esta métrica, por lo que se considera que sus EE están lo suficientemente defectuosos como para no tener la calidad requerida. Un segundo lugar le corresponde a la analista Yeni con una densidad de 4.3, realizando 23 EE en los que se detectaron 99 no-conformidades, se ubica en el segundo rango por lo que se considera que la calidad de sus EE es aceptable, el tercer lugar es para la analista Diana, realizando solamente 10 EE, detectándole 38 no-conformidades, representando una densidad de 3.80, se ubica en el segundo rango considerándose una calidad media en sus EE. También resalta la analista Yunexis, con 21 EE y 71 no-conformidades detectadas, obteniendo una densidad de 3.38, ubicándose también en el segundo rango, logrando una calidad media de sus EE. Sin embargo es notable la densidad de los analistas Reynaldo (0.73) y Susel (0.93), los cuales realizaron 37 EE y 15 EE, encontrándole 27 y 14 no-conformidades respectivamente, ambos se ubican en el primer rango, significando la menor cantidad de defectos en sus EE, por lo tanto son los analistas con mayor calidad en sus EE. La densidad promedio por analista es de 2.73.

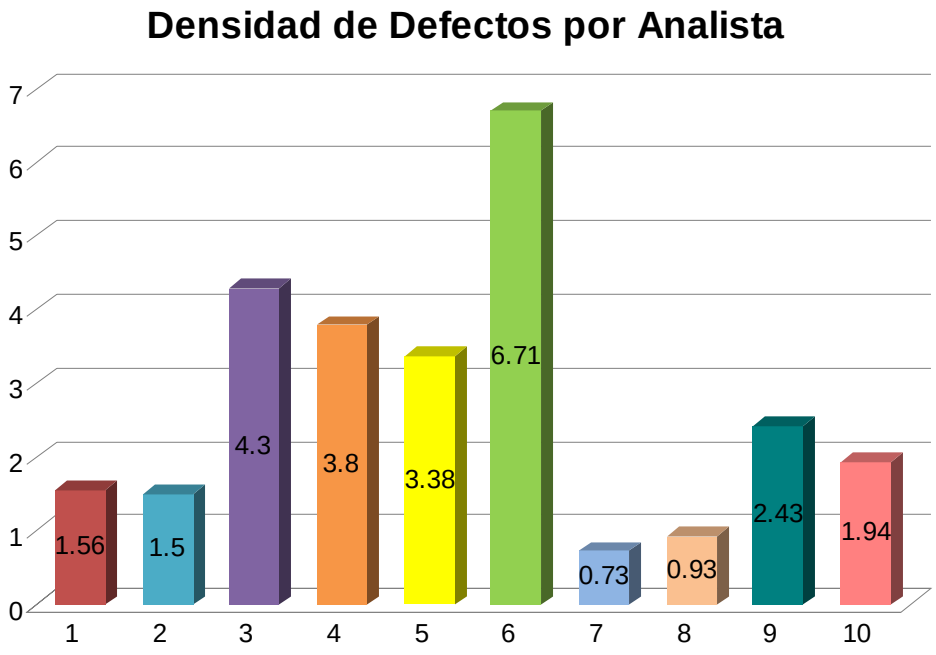

Figura 3. 4 Densidad de defectos por analista

3.4.5 Cantidad de EE revisados por Revisor

El siguiente gráfico (Figura 3.5) muestra la cantidad de EE revisados por los revisores que participaron en el proceso de revisión. Se observa que el revisor Yendi fue el que más EE tuvo su cargo, revisando 36 EE que representa un 17.65% del total de EE revisados. Después de él, se destacan los revisores Yudenia y Liudmila con 24 y 21 EE respectivamente, mientras que los revisores que menos EE revisaron son Carlos Manuel, Ángel Alberto, Maylen y Leonardo, el primero con solo dos EE y el resto con tres EE. La Tabla 3.4 muestra el orden en que aparecen los revisores en los gráficos que se presentan.

Tabla 3.5 Orden de aparición de los revisores en los gráficos

No	Revisor	No	Revisor
1	Liudmila	10	Maylen
2	Ángel Alberto	11	Haydée
3	Yinet	12	Abel Méndez
4	Liana	13	Erick Eduardo
5	Carlos Manuel	14	Yendi
6	Yainerys	15	Greilan
7	Yudenia	16	Jorge Amado
8	Pedro	17	Dairy
9	Pablo	18	Leonardo

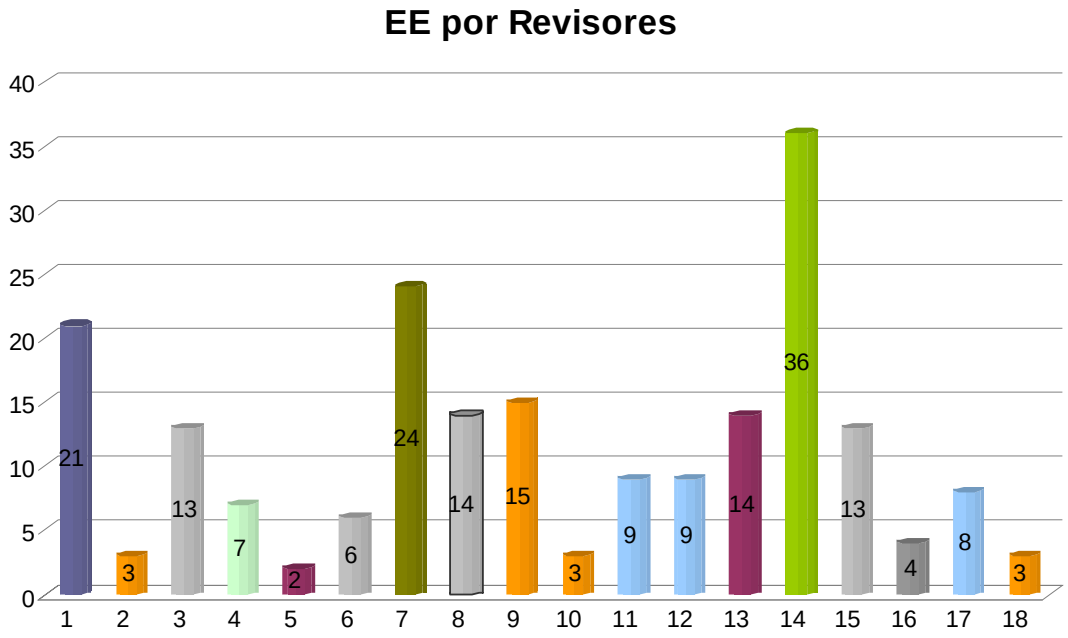

Figura 3. 5 Cantidad de EE revisados por revisor

3.4.6 Cantidad de defectos detectados por revisor

En este gráfico (Figura 3.6) se muestra la cantidad de defectos que detectó cada uno de los revisores que participaron en el proceso de revisión. Se puede deducir que la revisora que más defectos detectó es Yinet con 117 defectos representando el 20.56% del total de defectos detectados. Le sigue la revisora Liudmila con 97 defectos detectados representando el 17.05% del total. Es válido destacar la participación de la revisora Liana con 90 defectos detectados representando un 15.82% del total. Por otro lado, los revisores que menos defectos detectaron son Leonardo (2), Pablo (2), Carlos Manuel (3), Jorge Amado (4) y Erick Eduardo (5). Llama la atención que todos estos revisores son de sexo masculino. El promedio de defectos detectados por revisor es de 28.55 defectos.

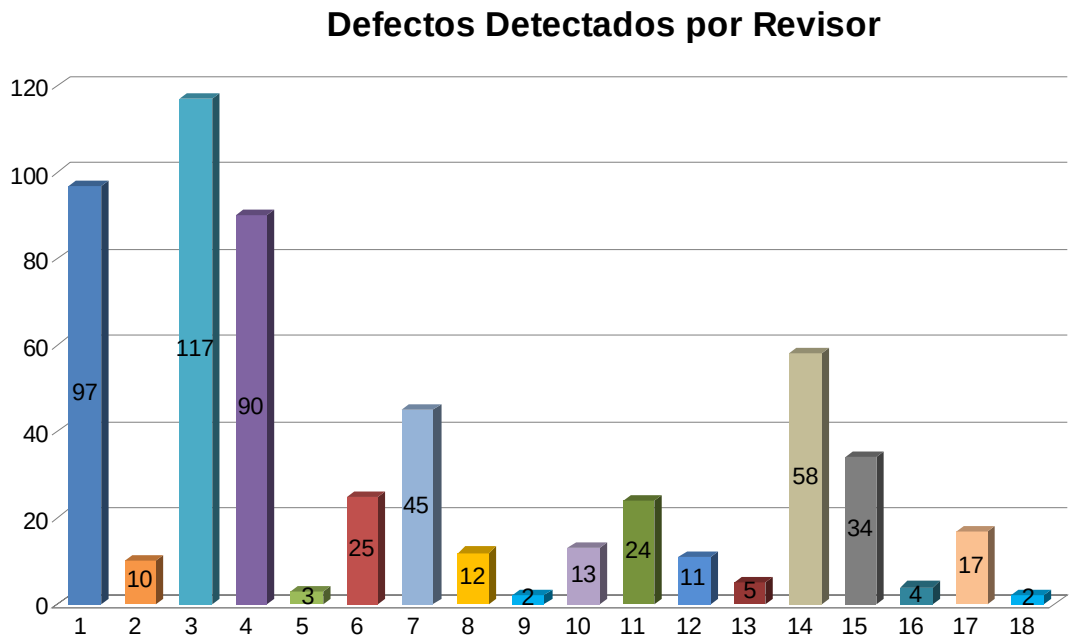

Figura 3. 6 Cantidad de defectos detectados por revisor

3.4.7 Densidad de defectos por revisor

A continuación el gráfico (Figura 3.7) muestra la densidad de defectos detectados por revisor. En este caso la interpretación de esta métrica difiere de la densidad de defectos por analista, pues mientras mayor sea el valor, se puede afirmar que la revisión a los EE fue más profunda. En este sentido, se destacan las

revisoras Liana con una densidad de 12.85, ubicándose en el primer rango establecido para esta métrica, por lo que su densidad es alta y sus revisiones fueron profundas logrando detectar la mayoría de los defectos presentes en los EE; le sigue Yinet con una densidad de 9, por lo que sus revisiones a pesar de que se ubican en el segundo rango, es una de las que más defectos detectó en sus EE, por lo que se considera que sus revisiones fueron profundas al igual que Liudmila con 4.62; por otro lado, Maylen fue de las que menos EE revisó, sin embargo fue bastante exhaustiva en sus revisiones, obteniendo una densidad de 4.33, finalmente se destacó la revisora Yainerys con 4.17. Llama la atención que todos los revisores que se destacan en este caso son de sexo femenino. Mientras que los revisores que menor densidad obtuvieron son: Pablo que revisó 15 EE detectando solamente 2 defectos, obteniendo una densidad de 0.13, Erick Eduardo que revisó 14 EE y detectó 5 defectos obteniendo una densidad de 0.36, Leonardo con una densidad de 0.67 y Pedro con 0.86. La densidad promedio es de 3.04.

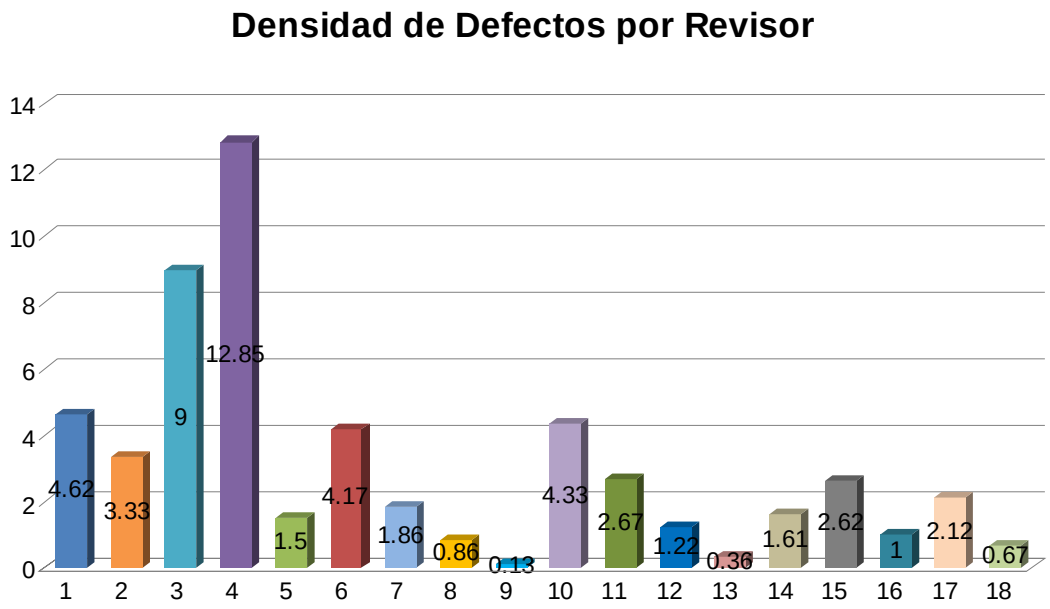

Figura 3. 7 Densidad de defectos por revisor

3.4.8 Clasificación de defectos

El siguiente gráfico (Figura 3.8) representa el por ciento de cada tipo de defecto. Los defectos se clasificaron en dos grandes grupos para poder saber hacia dónde enfocar las medidas necesarias con el

objetivo de mejorar el proceso de GR y el proceso de revisión con la toma de acciones correctivas a todos los entes que se encuentran involucrados. La clasificación se hizo teniendo en cuenta los defectos que cometieron los analistas, donde todos presentaron en alguna medida problemas de redacción y problemas técnicos. Del total de defectos detectados en todos los EE revisados, 344 se encuentra en el grupo de defectos por redacción, representando un 60%, mientras que 225 están en el grupo de defectos técnicos, representando el 40% restante, tal como muestra el gráfico. Estos resultados indican que se debe prestar especial atención a la redacción de los EE, valorando la posibilidad de dedicar un tiempo adicional en la planificación personal del analista para revisar sus propias especificaciones. Se considera que la causa que más ha influido en estos resultados fue el escaso tiempo y la carga de trabajo asignada en esa etapa.

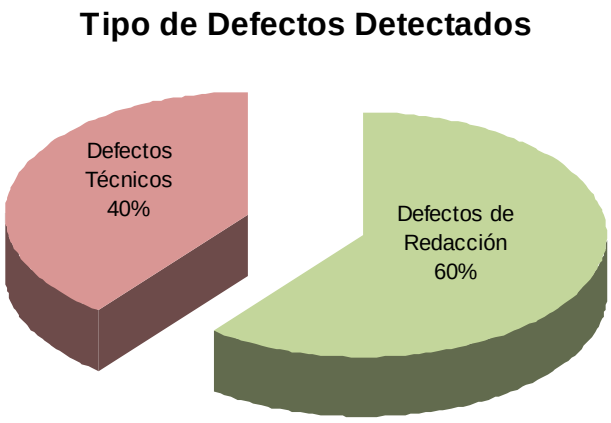

Figura 3. 8 Clasificación de defectos

3.4.9 Analistas con mayor densidad de defectos

El gráfico (Figura 3.9) que se muestra a continuación representa a los analistas con mayor densidad de defectos en sus EE con el objetivo de saber cuáles son los mayores problemas que presentaron, en este intento se analizaron los defectos que cometieron y se ubicaron en la clasificación establecida. Se puede concluir que el analista 1 (Miguel A) de sus 188 defectos 160 están ubicados en defectos de redacción, representando el 85.11% de sus defectos y los 28 (14.89%) restantes se deben a defectos técnicos, por lo que la mayor atención para este analista debe estar centrada en la redacción, ortografía y concordancia

de las ideas. En el mismo caso se encuentran las analistas Yeni, que de sus 99 defectos, 53 son de redacción y 46 técnicos y Diana con 38 defectos de los cuales 27 son de redacción y 11 técnicos. Por otro lado, la analista Yunexis presenta de sus 71 defectos, 31 de redacción y 40 técnicos, por lo que se debe enfocar en asuntos técnicos de especificación de EE.

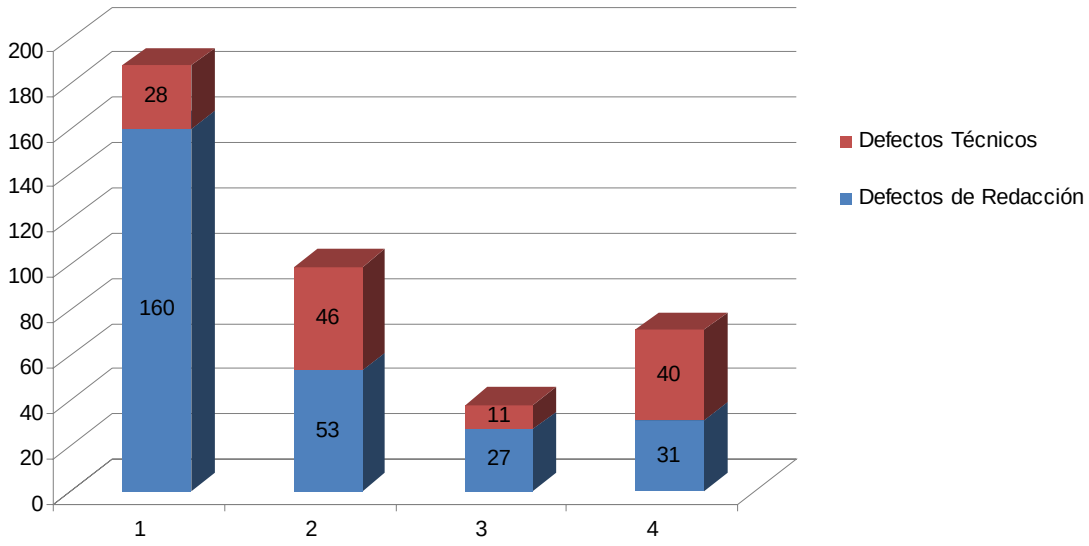

Figura 3. 9 Defectos de los analistas con mayor densidad de defectos

Considerando que la productividad promedio de la revisión (**PRP**) es de 7.29 y teniendo en cuenta que no se tienen valores anteriores de esta métrica en otros procesos o proyectos e incluso en otras revisiones del mismo artefacto, se puede concluir de forma general que la revisión fue productiva, ya que la cantidad total de EE revisados (TEER) es lo sumamente amplia con respecto al tiempo que se empleó para revisarlos.

La razón de realización promedio de los analistas (**RPPA**) es 5.1, esta métrica es un indicador de la cantidad de EE que realizó cada analista en sus procesos del total de EE realizados en el artefacto completo, teniendo en cuenta el tiempo que emplearon para su realización. A partir de este valor se pueden hacer planificaciones para futuros procesos y realizaciones de EE.

Por otro lado, el esfuerzo promedio por EE de los revisores (**EEER**) es de 0.56, significa que los revisores no tuvieron que realizar mucho esfuerzo para revisar los EE que tenían planificados.

Por último, la eficiencia de la revisión (**ER**) es de 1.43, por lo que se puede concluir que fue una revisión eficiente, considerando que se había planificado 40 hrs de trabajo y en 28 hrs se logró revisar todos los EE del PEN, con 12 hrs de diferencia.

3.5 Principales problemas detectados

Después de haber analizado los defectos que se detectaron en el PEN, se pueden enumerar un conjunto de problemas que están afectando el óptimo desarrollo de la GR en el proyecto CICPC, alguna de ellas son:

- Problemas de Redacción.
  - Errores ortográficos
  - Errores de concordancia
  - Errores de formato
- Problemas Técnicos.
- Empleo ineficiente del tiempo.
- Empleo ineficiente de los recursos.
- Desorganización en la forma de trabajar.
- Poca experiencia en el rol que desempeñan.
- Poca supervisión del trabajo que realizan los analistas.
- Muchas horas de trabajo diarias.
- Mala división del trabajo.

3.6 Posibles medidas correctivas

Alguna de las medidas que se recomiendan para superar estas deficiencias son las siguientes:

- Inculcar la inclusión de un tiempo adicional personal para las auto-revisiones.
- Aumentar el control de los responsables del proceso sobre el trabajo de los analistas.
- Aumentar la frecuencia de las revisiones por parte de los revisores.

- ♠ Mejorar la organización del trabajo para garantizar el cumplimiento de las responsabilidades de cada rol.
- ♠ Hacer las estimaciones más realistas para evitar cúmulos de trabajo y evitar la tensión de los analistas.
- ♠ Reconocer el esfuerzo empleado por los analistas con el objetivo de aumentar la motivación y la conciencia que los puede incentivar a realizar un trabajo con calidad.

3.7 Conclusiones del capítulo

En este capítulo se aplicó la fase de medición al proceso, haciendo un análisis valorativo de los resultados obtenidos, que permitió detectar las principales causas o deficiencias que se manifiestan actualmente en el proceso de GR del proyecto CICPC y a partir de ahí, trazar una línea base con las posibles medidas a tener en cuenta. Todas las métricas que se muestran en este apartado dan respuesta en alguna medida a las tres metas u objetivos señalados en el epígrafe 2.4.1. Para referirse al mejoramiento del proceso de revisión se empleó la métrica **ER**, donde se concluyó que fue eficiente, esto indica que se debe seguir trabajando de esa forma en futuras revisiones; para inferir sobre el control y seguimiento de los procesos de GR y Revisión se emplearon las métricas **PRP**, concluyendo que el personal involucrado con estos procesos cumplieron con el tiempo establecido, **DDP_A**, **DDP_R**, **PDEE** y **TEER** que reflejaron de forma general la buena calidad de ambos procesos, además para colegir con respecto a la planificación, se empleó la métrica **EEER**, demostrando que se hizo una buena planificación, teniendo en cuenta que hubo que realizar mucho esfuerzo para revisar los EE. Finalmente, se considera que siguiendo la misma línea de trabajo que se lleva hasta el momento y corrigiendo las deficiencias detectadas, es casi seguro lograr una ERS al terminar el proceso de GR con la calidad requerida.

Conclusiones

Al finalizar la presente investigación se concluye lo siguiente:

- Se demostró en esta investigación que para minimizar la complejidad y relatividad inherentes al concepto de calidad de los sistemas software, se hace necesaria la aplicación de modelos que permitan estimar la calidad del proceso y de productos software.
- Se demostró que no solo es posible “medir” la calidad, sino también “construir” la calidad durante el proceso de desarrollo, utilizando para ello las actividades de control y gestión de la calidad.
- Las definiciones de las características establecidas, tomando como base las características de calidad del estándar ISO/IEC 9126, vinculadas con la calidad de la especificación propician una buena base teórica para abordar de forma efectiva la medición de la ERS.
- Se diseñaron plantillas de mediciones que ayudan a registrar los datos que se utilizan en el cálculo de las métricas.
- Se hizo una propuesta de métricas incluidas en un procedimiento de medición de calidad en la GR, pero que pueden ser extendidas a otros procesos o modelos y que permiten hacer un análisis del proceso de revisiones así como la toma de medidas oportunas durante la propia ejecución del proceso de Gestión de Requisitos.
- Se propicia el mejoramiento continuo del proceso al poder contar con un conjunto de métricas que permitan medir la efectividad de las revisiones y del proceso de GR, a partir de los datos recogidos como resultado de los elementos de especificación revisados.
- El procedimiento RPQ abarca las actividades necesarias para darle un control y seguimiento a la Gestión de requisitos, proponiendo dos enfoques cuantitativos dirigidos a la calidad del proceso y del producto especificación.
- Se aplicó una de las fases del procedimiento propuesto en un proyecto mostrando la forma en que se pueden identificar los problemas y determinar posibles soluciones.

Recomendaciones

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

- ♠ Aplicar el procedimiento RPQ a otros proyectos de la UCI para obtener los beneficios del mismo.
- ♠ Desarrollar una herramienta capaz de registrar y calcular las métricas propuestas para la medición de la calidad en la Gestión de Requisitos.
- ♠ Implantar un repositorio para almacenar las métricas y datos referentes de todos los proyectos que se realicen en la Facultad 8 con el propósito de emplear la documentación en análisis comparativos que ayuden al mejoramiento y predicción de comportamientos en futuros procesos de desarrollo de un software, además que recopilen la experiencia en otros proyectos de la utilización de modelos de calidad con el propósito de crear centros de experiencia para medir la calidad de los sistemas software.
- ♠ Analizar la inclusión del factor de calidad (QF) como parte de la evaluación de la calidad del producto, analizando las características que están presentes en una especificación de requisitos según las tres perspectivas indispensables en el desarrollo de un software que incluyen a los usuarios, desarrolladores y administradores, representando la dimensión social, técnica y administrativa respectivamente.
- ♠ Identificar otras métricas de calidad que se pueden emplear en aras de medir la calidad de la especificación, basándose en la clasificación de características de calidad establecidas.

Referencias bibliográficas

Documentos Impresos	
[Cruz, 2005]	Cruz, Haydée Mª. GREQOFF: *Modelo para la Gestión de Requisitos en el desarrollo "Offshore".* Tesis (Master en Informática Aplicada). Ciudad de la Habana, Cuba, Instituto Superior Politécnico José Antonio Echeverría, Centro de estudios de Ingeniería y Sistemas, 2005. 124 p.
[Humphrey, 2001]	Watts S, Humphrey. *Introducción al Proceso de Software Personal*. Primera Edición en español, Pearson Educación, 2001.
[Pressman, 2002]	Pressman, Roger S. *Ingeniería de Software, un enfoque práctico.* Quinta edición. McGraw-Hill, 2002.
Documentos Electrónicos	
[Aguillón, 2007]	Aguillón, Esperanza R. *Métricas para la Gestión de Proyectos de Software* [en línea]. Instituto Tecnológico de Morelia, 2007 [visitado: 20 de Mayo de 2007]. Disponible en: http://sistemas.itlp.edu.mx/memoriaSIC03/metricas.pdf
[Andújar, 2004]	Andújar U, Juan Antonio. *Métricas de Calidad para Proyectos de Software* [en línea]. 2004 [visitado: 31 de Octubre de 2006]. Disponible en: http://www.pmies.org/ponencias/2004/ponencia%20Juan%20A%20Andujar.pdf
[Barceló, 2003]	Barceló, Miguel. *Estimación de costes de un proyecto informático. Métricas de productividad y modelos de estimación de costes* [en línea]. 2006 [visitado: 16 de Mayo de 2007]. Disponible en: http://bibliotecnica.upc.es/cast/fenixdoc/Invest.asp?Id=0002087&UE=723
[Buglione & Abran, 1999]	Buglione, Luigi y Abran, Alain. *A Quality Factor for Software* [en línea]. 1999 [visitado: 22 de Marzo de 2007]. Disponible en: www.lrgl.uqam.ca/publications/pdf/408.pdf
[de Antonio, 2002]	de Antonio J, Angélica. *Gestión, Control y Garantía de la Calidad del Software* [en línea]. 2002 [visitado: 17 de Marzo de 2007]. Disponible en: http://www.inf.uach.cl/rvega/asignaturas/info265/G_Calidad.pdf
[Delgado, 2005]	Delgado, Martha D., Álvarez, Sofía & Rosete, Alejandro. Una Propuesta de Introducción de las Revisiones en el Proceso de Desarrollo de Software. *Revista Investigación Operacional*, 2005 Vol. 26, No. 2, p (99 - 113) Disponible en: http://www.dict.uh.cu/Revistas/IO2005/Vol%2026%20No.2/IO%2026205-1.doc
[Durán, 2000]	Durán Toro, Amador. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas*

de Información. Memoria de la tesis doctoral dirigida por el doctor José Miguel Toro Bonilla y desarrollada por Amador Durán Toro para optar al grado de Doctor en Informática por la Universidad de Sevilla. Septiembre de 2000 [visitado: 15 de Enero de 2007]. Disponible en: http://www.lsi.us.es/~amador/publicaciones/tesis.pdf.zip

[Escalona & Koch, 2002] Escalona, María J y Koch, Nora. *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo* [en línea]. 2002 [visitado: 17 de Marzo de 2007]. Disponible en: http://lsiweb.lsi.us.es/docs/informes/LSI-2002-4.pdf

[Fuente, 2001] Fuente, Aquilino J. *Necesidad de Sistemas Formales de Métricas para Proyectos de Software OO* [en línea]. Universidad de Oviedo, 2001 [visitado: 14 de Diciembre de 2006]. Disponible en: http://www.di.uniovi.es/~aquilino/ficheros/articulos/Metricas/Sevilla-JJOO-1997.pdf

[Goodman, 2004] Goodman, Paul. *Software Metrics-Best Practices for Successful IT Management*. ISBN: 1-931332-26-6. Rothstein Associates Inc., Publisher. 2004. P 264. [visitado: 25 de Octubre de 2006] Disponible en: http://sharespot.flazx.net/download/1931332266.zip.htm

[González, 2001] González, Doria H. *Las Métricas de Software y su Uso en la Región*. Tesis Licenciatura. Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas-Puebla. Mayo, 2001.

[IBM Rational RequisitePro, 2006] IBM Software Group. *Gestión de requisitos a lo largo de todo el ciclo de desarrollo* [en línea]. 2006 [visitado: 31 de Octubre de 2006]. Disponible en: http://www.calidaddelsoftware.com/eventos/SoloRqstos2006//Solo%20Requisitos%202006/Rational%20RequisitePro.pdf

[InformeSEI, 2004] Informe de investigación del SEI (Software Engineering Institute - Instituto de Ingeniería del Software). 2004

[Kan, 2002] Kan, Stephen H. *Metrics and Models in Software Quality Engineering*. Segunda Edición. Addison Wesley. ISBN: 0-201-72915-6, p 560. 2002. [visitado: 02 de Abril de 2007]. Disponible en: http://www.flazx.com/ebook5838.php

[Laird & Brennan, 2006] Laird, Linda M y Brennan, M. Carol. *Software Measurement and Estimation: A Practical Approach*. ISBN 0-471-67622-5. John Wiley & Sons, Inc., Hoboken, New Jersey. 2006. P 276. [visitado: 25 de Octubre de 2006] Disponible en: http://sharespot.flazx.net/download/0471676225.zip.htm

[Letelier, 2003] Letelier, Patricio. *Proceso de desarrollo de software* [en línea]. 2003 [visitado: 27 de Enero de 2007]. Disponible en:

https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20Proceso%20de%20Desarrollo%20de%20SW.doc

[Monzón, 2002] Monzón, Antonio. *Calidad de la Especificación: ¿Se pueden medir los Requisitos?* [en línea] TCP Sistemas e Ingeniería, 2002 [visitado: 27 de Octubre de 2006]. Disponible en: http://irqaonline.fileburst.com/docs/papers/Calidad_de_la_Especificacion.pdf

[Munson, 2003] Munson, John C. *Software Engineering Measurement*. ISBN: 0849315034. Auerbach Publications, 2003. P 443.

[Olsina, 1999] Olsina, Luis A. Ingeniería de Software en la Web. *Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web* [en línea]. Tesis Doctoral. Facultad de Ciencias Exactas, Universidad Nacional de La Plata – Argentina, 1999. 274p. [visitado: 25 de Noviembre 2006]. Disponible en: http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Web-site_QEM_VF.pdf

[Orantes & Botello, 2006] Orantes, Sandra y Botello, Alejandro. *Necesidad de la Medición de la Calidad de Software de los Sistemas Multiagentes* [en línea]. 2006 [visitado: 27 de Octubre de 2006]. Disponible en: http://enc.smcc.org.mx/proceedings/talleres/TALLER-SIS06/submission_9.pdf

[Palacio, 2006] Compendio de Ingeniería del Software II (CIS) [en línea]. Rev. 0.03 España. Palacio, Juan. Junio / 2006. [Visitado: 28 de abril de 2007]. Disponible en: http://hornet.ls.fi.upm.es/DSP/Lecciones/lect06-9-enero-2007.pdf

[Palomino, 2006] Palomino, Miguel. *¿Se pierden sus requisitos por el camino? Introducción a OptimalTrace* [en línea]. 2006 [visitado: 31 de Octubre de 2006]. Disponible en: http://www.calidaddelsoftware.com/eventos/SoloRqstos2006//Solo%20Requisitos%202006/OptimalTrace.pdf

[Perez, 2004] Perez G, Otoniel. *Métricas, Estimación y Planificación en Proyectos de Software* [en línea]. Universidad de Guadalajara. 2004 [visitado: 05 de Octubre de 2006]. Disponible en: http://www.willydev.net/Descargas/WillyDEV_PlaneaSoftware.Pdf

[RequisitePro, 2003] Rational RequisitePro Online Help, Rational Software Corporation, 2003.

[Rossi, 2003] Rossi, Bibiana. *Métricas de Software* [en línea]. 2003 Reportes Técnicos [visitado: 18 de Mayo de 2007]. Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/ROSSI-METRICAS.pdf

[RUP, 2003] Proceso Unificado de Rational. IBM Rational Suite 2003

[Sumano, 2000] Sumano, Mª de los Ángeles. *Análisis de Requerimientos de Software. Estado del Arte* [en

línea]. 2000 [visitado: 30 de Enero de 2007]. Disponible en: http://www.geocities.com/diegolp/ingsof/requerimientos.pdf

Estándares	
[IEEEComputer, 2004]	IEEE Computer Society. A Compilation of Software Engineering Terms from Existing Sources. Certified Software Development Professional (*csdp)* http://www.computer.org/certification/csdpprep/Glossary.htm
[IEEE 610, 1990]	Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Computer Dictionary. Software Engineering Terms, 1990. ISBN: 1559370793, 1991. 218 p.
[IEEE 729, 1983]	Instituto de Ingenieros Eléctricos y Electrónicos. Glossary of Software Engineering Terminology – Redesignated as IEEE 610.12, 1983.
[IEEE 830, 1998]	Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Recommended Practice for Software Requirements Specifications. PDF: ISBN 0-7381-0448-5, SS94654, 1998. 38 p.
[IEEE 1061, 1998]	Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Standard for a Software Quality Metrics Methodology, 1998. ISBN 0-7381-1510-X SS94706. 26 p.
[ISO/TC 176/SC1N322, 2007]	Organización Internacional para la Estandarización. Concepts and Terminology. Revision of ISO 9000 Quality management systems - Fundamentals and vocabulary Preliminary working draft. 2007. 45 p.
[ISO/IEC 9126-2, 2003]	Organización Internacional para la Estandarización. ISO/IEC TR 9126-2:2003 Software engineering – Product quality – Part 2: External metrics. CH-1211 Ginebra, Suiza, 2003. 94 p.
[ISO/IEC 9126-3, 2003]	Organización Internacional para la Estandarización. ISO/IEC TR 9126-3:2003 Software engineering – Product quality – Part 3: Internal metrics. CH-1211 Ginebra, Suiza, 2003. 70 p.
[NC-ISO/IEC 9126-1, 2005]	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. 33 p.

Bibliografía

Documentos Impresos	
[Cruz, 2005]	Cruz, Haydée Mª. GREQOFF: *Modelo para la Gestión de Requisitos en el desarrollo "Offshore".* Tesis (Master en Informática Aplicada). Ciudad de la Habana, Cuba, Instituto Superior Politécnico José Antonio Echeverría, Centro de estudios de Ingeniería y Sistemas, 2005. 124 p.
[Humphrey, 2001]	Watts S, Humphrey. *Introducción al Proceso de Software Personal*. Primera Edición en español, Pearson Educación, 2001.
[Larman, 1999]	Larman, Craig. UML y Patrones. *Introducción al análisis y diseño orientado a objetos*. Tomo I, II, III. Prentice Hall, Inc. México, 1999. p 503.
[Pressman, 2002]	Pressman, Roger S. *Ingeniería de Software, un enfoque práctico.* Quinta edición. McGraw-Hill, 2002.
Documentos Electrónicos	
[Aguillón, 2007]	Aguillón, Esperanza R. *Métricas para la Gestión de Proyectos de Software* [en línea]. Instituto Tecnológico de Morelia, 2007 [visitado: 20 de Mayo de 2007]. Disponible en: http://sistemas.itlp.edu.mx/memoriaSIC03/metricas.pdf
[Fernández & Monzón, 2000]	Fernández, José L. & Monzón, Antonio. *A Metamodel and a Tool for Software Requirements Management* [en línea]. Reliable Software Technologies - Ada - Europe 2000, Postdam, Berlin, Germany. 2000 [visitado: 22 de Mayo de 2007]. Disponible en: http://www.programmingresearch.com/pdfs/tcp_docs/A_metamodel_and_a_tool.pdf

Otros Sitios visitados:

http://www.ceis.cl/Gestacion/Gestacion.htm (5.10.2006)

http://www.uml.org/ (7.5.2006)

http://standishgroup.com/ (5.10.2006)

http://perso.club-internet.fr/brouardf/SGBDRmerise.htm (7.5.2007)

http://www.map.es/csi/metrica3/ (7.5.2007)

http://www.rational.com/products/rup/index.jsp (7.5.2006)

http://www.softwareqatest.com/ (18.01.2007)

http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html (7.5.2007)

http://portal.newman.wa.edu.au/technology/12infsys/html/dfdnotes.doc (29.8.2006)

Glosario de Términos y Siglas

Términos

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

Atributo de calidad*:* Una característica del software, o un término genérico aplicado a las características de calidad, sub- características de calidad.

Control de Calidad: Actividades para evaluar la calidad de los productos desarrollados.

Defecto: Consecuencia de un error. Incumplimiento de un requisito asociado a un uso previsto o especificado.

Ente: Se refiere a los procesos, productos y servicios.

Error: Acción humana durante el proceso de desarrollo que produce un defecto.

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

Gestión de Calidad: Determinación y aplicación de las políticas de calidad de la empresa (objetivos y directrices generales).

Indicadores: Métricas que se calculan a partir del resultado de otras métricas calculadas.

Métrica de calidad del software: Una función cuyas entradas son los datos del software y que la salida es un solo valor numérico que puede interpretarse como el grado en que el software posee un atributo dado que afecta su calidad.

NOTA: Esta definición difiere de la definición de métrica de calidad encontrada en IEEE Std 610.12-1990.

Métrica del proceso: Una métrica usada para medir características de los métodos, técnicas, y herramientas empleadas en el desarrollo, implementación, y mantenimiento el sistema del software.

Métrica del producto: Una métrica usada para medir las características de cualquier producto intermedio o final del proceso de desarrollo del software.

Métricas directas: Métricas que se obtienen a partir de la observación de la manifestación directa de datos.

Métricas indirectas: Métricas que se calculan a partir de otras.

Modelamiento del negocio: No es más que comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema; comprender los problemas actuales de la organización e identificar

las mejoras potenciales; asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización y derivar los requisitos del sistema que va a soportar la organización.

Modelos de Calidad: En los modelos de calidad, la calidad se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. La calidad es un concepto que se deriva de un conjunto de sub-conceptos.

No-conformidad: Incumplimiento con los requisitos establecido en una lista de chequeo.

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

Requisito: Condición o cualidad que debe cumplir alguien o algo.

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

Validación de los requisitos: Actividad para detectar omisiones o ambigüedades en los requisitos.

Siglas

ACM - Association for Computing Machinery - Asociación para la Maquinaria de Computación

CASE - Computer-Aided Software Engineering - Ingeniería de Software Asistida por Ordenador

C - Característica de Calidad

CC - Criterios de Calidad

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

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

CU - Caso de Uso del Sistema

CUN - Caso de Uso del Negocio

EC - Elementos de Configuración

EE - Elemento de Especificación

EFQM - European Foundation for Quality Management - Fundación Europea para la Gestión de la Calidad

ERS - Especificación de Requisitos del Software

FC - Factor de Calidad

FCM - Factors-Criteria-Metrics - Factor-Criterio-Métrica

FT - Flujo de Trabajo

GQM - Goal-Question-Metric - Meta-Pregunta-Métrica

GR - Gestión de Requisitos

ID - Identificador de la variable

IEC - International Electrotechnical Commission - Comisión Electrotécnica Internacional

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

IR - Ingeniería de Requisitos

ISO - International Organization for Standardization - Organización Internacional para la Estandarización

JAD - Joint Application Development - Desarrollo Conjunto de Aplicaciones

LCD - Líneas de Código

MC - Métricas de Calidad

MEPS - Mejora Estadística de Proceso del Software

PEN - Procesos Elementales del Negocio

PR - Proceso de Revisión

PSM - Practical Software and Systems Measurement - Mediciones Prácticas del Software y Sistemas

PSP - Personal Software Process - Proceso Personal del Software

QF - Quality Factor - Factor de Calidad

RPQ - Requirements Process Quality - Calidad en el Proceso Gestión de Requisitos

RUP - Rational Unified Process - Proceso Unificado del Rational

SC - Sub-característica de Calidad

SEI - Software Engineering Institute - Instituto de Ingeniería del Software

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

SQM - Software Quality Management - Gestión de Calidad del Software

SquaRE - Security Quality Requirements Engineering - Ingeniería de Requisitos de Calidad de Seguridad

SRS - Software/System-Requirements-Specification - Especificación de Requisitos del Software

SW - Software

SWEBOK - Software Engineering Body of Knowledge - Cuerpo de Conocimientos de Ingeniería de Software

UCI - Universidad de las Ciencias Informáticas

VD - Variables Dependientes

VI - Variables Independientes