Aseguramiento de la calidad en el diseño del software

Asesor:

Rafael David Rincón Bermúdez

Profesor

Departamento De Informática y Sistemas

Autor:

Jaime Eduardo Marulanda López

Universidad EAFIT

Escuela De Ingeniera

Ingeniería De Sistemas

Medellín

2014

Tabla de contenido

Lista de imágenes

Lista de tablas

Lista de formatos

Glosario de términos

- **Artefacto:** Producto tangible resultante del proceso de desarrollo de software.
- **Atributo:** Propiedad inherente de una entidad que puede distinguirse cuantitativa o cualitativamente ya sea manual o automáticamente.
- **Benchmark:** Técnica utilizada para medir el rendimiento de un sistema o componente del mismo, frecuentemente en comparación con el que se refiere específicamente a la acción de ejecutar un benchmark. La palabra benchmark es un anglicismo traducible al español como comparativa.

  Más formalmente puede entenderse que un benchmark es el resultado de la ejecución de un programa informático o un conjunto de programas en una máquina, con el objetivo de estimar el rendimiento de un elemento concreto, y poder comparar los resultados con máquinas similares.
- **Calidad:** Capacidad de un producto, servicio o proceso para proporcional el valor deseado.
- **Calidad interna:** Capacidad de un conjunto estático de atributos para satisfacer las necesidades declaradas e implícitas de un producto software bajo ciertas condiciones especificadas.
- **Calidad externa:** Capacidad de un producto software para desarrollar el comportamiento de un sistema de forma que satisfaga las necesidades declaradas e implícitas de un sistema utilizado bajo ciertas condiciones especificadas.
- **Calidad en uso:** Grado en que un producto satisface objetivos con efectividad, seguridad, satisfacción y productividad.

- **Caso de uso:** Es una descripción de una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.
- **Código fuente:** Es un conjunto de líneas de texto que indican las instrucciones que debe seguir el computador para ejecutar un programa. Por lo tanto, en el código fuente de un programa está escrito por completo su funcionamiento.
- **DFD (Diagrama de flujo de datos):** Representación gráfica del flujo de datos a través de un sistema de información. Un diagrama de flujo de datos también se puede utilizar para la visualización de procesamiento de datos (diseño estructurado). Es una práctica común para un diseñador dibujar un contexto a nivel de DFD que primero muestra la interacción entre el sistema y las entidades externas.
- **Diagrama de casos de uso:** Especifican la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

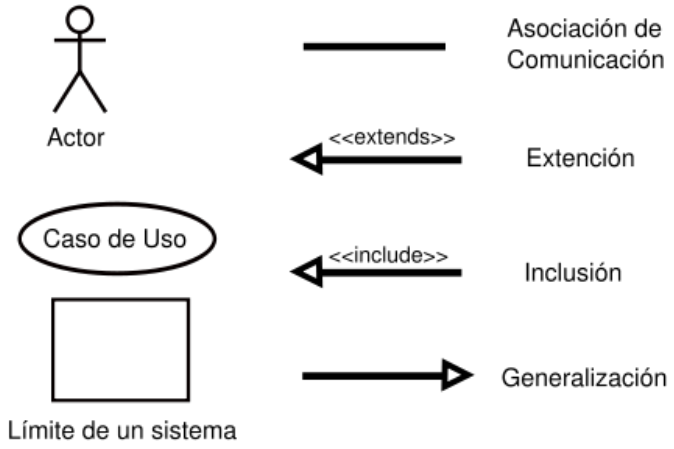

Imagen 1 Notación de casos de uso. (Fuente: Utilización de UML en ingeniería del software con objetos y componentes. Perdita Stevens, Rob Pooley. Addison Wesley, 2002)

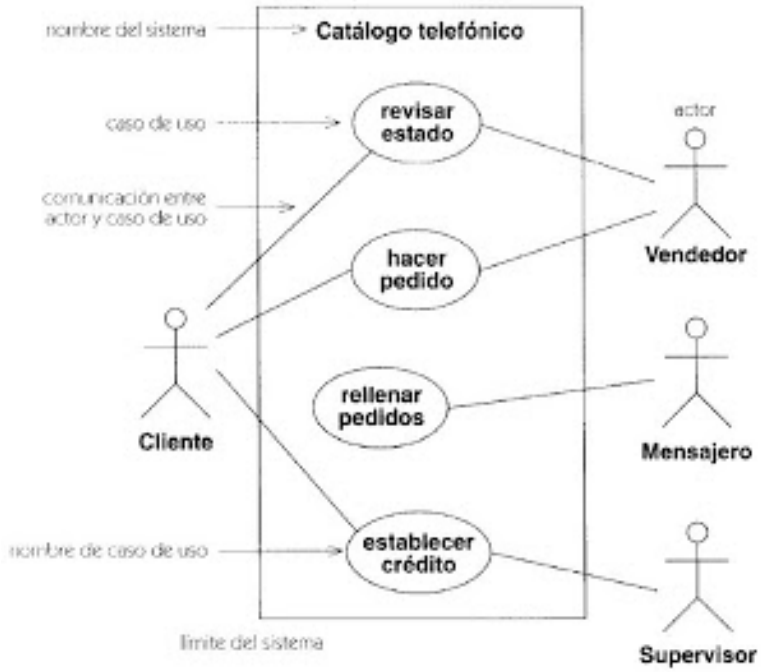

Imagen 2 Ejemplo diagrama caso de uso (Fuente: Jacobson, 1994)

- Diseño de arquitectura: Modelo de la estructura global del sistema.

- **IEC (International Electrotechnical Commission):** Organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas.
- **IEEE (Institute of Electrical and Electronics Engineering):** Asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

  Asociación internacional sin ánimo de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, ingenieros en informática, matemáticos aplicados, ingenieros biomédicos, ingenieros en telecomunicación e ingenieros en Mecatrónica.
- **ISO (International Organization Standarization):** Organismo encargado de promover el desarrollo de normas internacionales de fabricación (tanto de productos como de servicios), comercio y comunicación para todas las ramas industriales, con excepción de la Eléctrica y la Electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones (públicas o privadas) a nivel internacional.
- **Proceso:** Conjunto de procedimientos relacionados o que interactúan entre si para generar un producto o servicio.
- **Producto:** Servicio y herramientas tecnológicas.
- **Servicio:** Ejecución de actividades, trabajos o tareas relacionadas con un producto de software, tales como su desarrollo, operación y mantenimiento.
- **SQuaRE (Software product quality requirements and evaluation):** Estándar que tiene como objetivo organizar, enriquecer y unificar dos procesos principales: la especificación de requisitos de calidad del software y evaluación de la calidad del software, soportada por el proceso de medición de calidad del software.

La serie ISO/IEC 25000:2005 reemplaza a dos estándares relacionados: ISO/IEC 9126 (Software Product Quality) e ISO/IEC 14598 (Software Product Evaluation).

- **Stakeholder:** Término inglés utilizado por primera vez por R. E. Freeman en su obra: "Strategic Management: A Stakeholder Approach" (Pitman, 1984), para referirse a «quienes pueden afectar o son afectados por las actividades de una empresa».

  Estos grupos son los públicos interesados o el entorno interesado ("stakeholders"), que según Freeman, deben ser considerados como un elemento esencial en la planificación estratégica de los negocios.

  Los stakeholders de un sistema también son llamados interesados, y que necesitan una solución.

  Desde el punto de vista del desarrollo de sistemas, un "Stakeholder" es aquella persona o entidad que está interesada en la realización de un proyecto o tarea, auspiciando el mismo, ya sea mediante su poder de decisión y/o de financiamiento o a través de su propio esfuerzo.

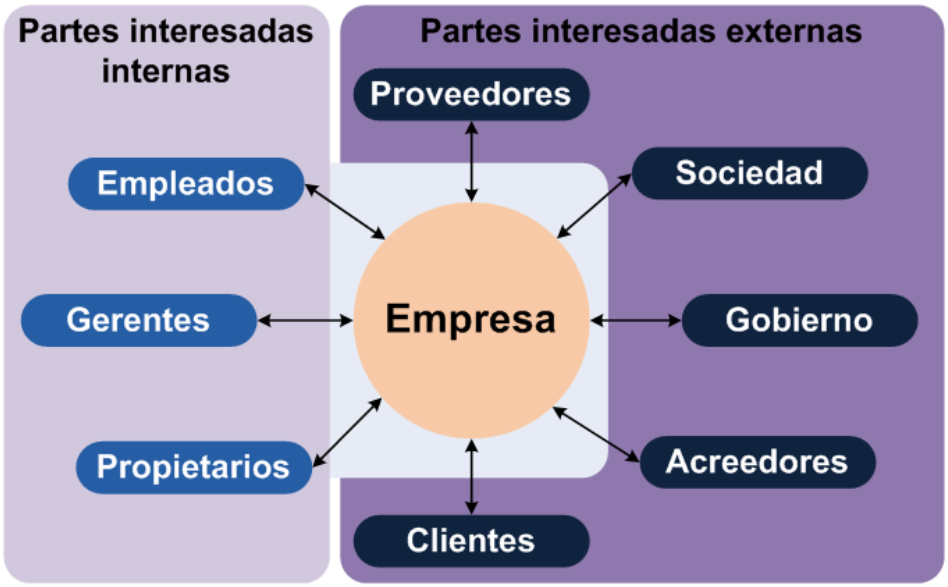

Imagen 3: Partes interesadas típicas de una empresa. (Fuente: Grochim, 2008)

- **Usuario:** individuo o grupo que interactúa con un sistema o los beneficios de un sistema durante su utilización.
- **Validación:** confirmación a través de la revisión objetiva, que los requisitos para un uso específico y previsto de una aplicación se hayan cumplido.

1 Introducción

“Se habla mucho del gran invento que ha sido el computador y toda su gran evolución en el gran mundo de la tecnología, pero dentro de este mundo tecnológico está el software, que nadie se esperaría que se convirtiera en una tecnología indispensable en casi todos los ámbitos que rodea al ser humano; no se esperaba que el software fuera ese gran “catapultador” de esta revolución de las nuevas tecnologías. Pero como consecuencia de estos avances, no se esperaba que a medida de que la importancia del software creciera, también se incrementaran los requisitos de normas y labores que exigen una alta calidad en todos estos productos”. (Pressman, 2005).

Respecto a la calidad del software, en la literatura se proponen varias definiciones, siendo una de ellas la que propone (Pressman, 2005): es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados, y de las características implícitas que se espera de todo software desarrollado profesionalmente.

En esta definición se resalta que los estándares de calidad definen un criterio de desarrollo que guía la forma como se aplica la Ingeniería del software; por lo tanto, si no se siguen esos criterios se incrementa la posibilidad de que el producto final no sea de calidad.

También se entiende que el estándar de calidad del software consiste en reunir todas las actividades y funciones, de tal forma que ninguna de ellas esté subordinada a las otras y que cada una planee, controle y ejecute de un modo formal y sistemático.

El desarrollo de un sistema de software va de la mano con la calidad de su diseño, el cual debe estar acompañado de unos requisitos que cumplan con la necesidad de los stakeholders.
Un sistema de software se verifica mediante pruebas (unitarias, integración, regresión, de humo, de sistemas, desempeño, carga, estrés, recuperación, entre otras) que garanticen que se cumple con los atributos funcionales (adecuación, precisión, conformidad, interoperabilidad, y seguridad) y no funcionales (fiabilidad, usabilidad, eficiencia, mantenibilidad, y portabilidad) al cien por ciento. La calidad no es un simple tema que está de moda, más bien es un factor importante, ya que los clientes y usuarios cada día son más selectivos y exigentes con sus necesidades. Como dice (Zavala. 2004, p.13) " el software de una organización se convierte de facto en un activo más valioso y que sin embargo, normalmente no se considera como tal".

Para que se alcance la calidad en el desarrollo de un sistema de software es necesario la utilización de metodologías o procedimientos estándares, cada metodología (RAD, Scrum, RUP, XP, Agile, entre otras) tiene en mayor o menor medida su propio enfoque (modelo en cascada, modelo de prototipos, modelo incremental, modelo en espiral, entre otros) los cuales proporcionan un marco de trabajo que permite estructurar, planear y controlar el proceso de desarrollo. Si no se sigue ninguna metodología siempre habrá falta de calidad.

Con el objetivo de alcázar los atributos funcionales y no funcionales de calidad, es importante seguir las diferentes etapas de la metodología de desarrollo de software seleccionada. Cada metodología indicara cuales etapas son necesarias para cada proyecto, permitiendo alcanzar un producto de calidad, desde su inicio, desarrollo, verificación y culminación.

Este trabajo de investigación pretende revisar la familia de estándares de calidad ISO/IEC 25000 (***Software engineering -- Software product quality requirements and evaluation (SQuaRE) -- Guide to SQuaRE.)***, para aplicarlo a la fase o etapa de Diseño en el desarrollo de software. Es importante aplicar el proceso de calidad en todos los estados de evolución (especificaciones, diseño, código, etc.,) del desarrollo del software, ya que mientras una inconsistencia sea detectada desde el inicio, a través de la prevención de errores y detección de defectos, tendrá menos costos que corregirla al final del proceso.

Para esta investigación la fase donde se centrará toda la atención es la de diseño, la cual involucra un proceso creativo de transformación del problema en una solución a través de la implementación de los requisitos contenidos en el modelo de análisis; es esta fase la que permite que los requisitos del cliente se puedan materializar en el producto del software finalizado; además es la base y núcleo fundamental para los pasos siguientes del desarrollo, permitiendo evaluar la calidad del desarrollo del proyecto por medio de los elementos y representaciones del mismo y con un conjunto de revisiones técnicas antes de continuar con los siguientes pasos.

La metodología propuesta en esta investigación se podrá implementar en las diferentes metodologías de desarrollo de software existentes (RAD, Scrum, RUP, XP, Agile, entre otras), esto debido a que en cualquiera de estas siempre existirá una fase de diseño.

Pero, ¿qué es Diseño de Software?

“ A principios de la década 1990, Mithch Kapor, el creador de Lotus 1-2-3, presentó un “*Manifiesto sobre el diseño del software*” en Dr. Dobbs Journal. Ahí afirmaba (Pressman, 2002. Pag.245)”:

> ¿Qué es diseño? Es el lugar en donde una persona se puede parar con un pie en dos mundos –el mundo de la tecnología y el de la gente y los propósitos humanos e intenta unirlos-…

Ahora bien. ¿Qué es calidad del software?. " Es la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente" (Pressman, 2005). Se puede decir que la calidad es una disciplina que debe estar inmersa en el desarrollo del software, y para esta investigación se enfocará en la fase del diseño con los estándares y metodologías de la familia de estándares ISO/IEC 25000. Esta norma propone un conjunto de características, subcaracterísticas y atributos para descomponer la calidad de un producto software. Las características (Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Seguridad, Interoperabilidad y Portabilidad), a su vez se dividen en subcaracterísticas. Se pretende asegurar la calidad en el diseño y finalmente repercutirá en la entrega de un producto software construido con calidad.

2 Planteamiento del problema

En los primeros años de desarrollo de las computadoras lo normal era que se le prestara mayor atención al hardware y este fuera de propósito general. Por otra parte, el software se diseñaba a medida para cada necesidad y tenía una distribución relativamente pequeña. La mayoría del software se desarrollaba y era utilizado por la misma persona u organización, la cual lo escribía, ejecutaba y si fallaba, lo depuraba. Debido a este entorno personalizado del software, el diseño era un proceso implícito, realizado por un limitado número de personas y por lo general, no realizaban ningún tipo de documentación. Adicionalmente, en estos primeros años la programación se veía como un “arte”, existían pocos métodos formales y no eran muy conocidos, por lo que muy pocas personas los aplicaban. El programador aprendía normalmente su oficio mediante prueba y error.

Hoy en día, los costos en el desarrollo de un sistema han cambiado drásticamente. El software, en lugar del hardware, es normalmente el elemento principal del costo.

La evolución en el diseño de software es un proceso continuo que se ha ido produciendo durante los últimos años. Los primeros trabajos sobre diseño se centraron en los criterios para el desarrollo de programas modulares y los métodos para mejorar la arquitectura del software de una manera descendente. Los aspectos procedimentales de la definición del diseño evolucionaron hacia una filosofía denominada programación estructurada. Posteriores trabajos propusieron métodos para la traducción del flujo de datos o de la estructura de los datos en una definición de diseño. Nuevos enfoques para el diseño proponen un método orientado a objetos para la obtención del diseño.

Cada metodología de diseño de software introduce heurísticas y notaciones propias, así como una visión algo particular de lo que caracteriza a la calidad del diseño.

El diseño puede definirse como el “proceso iterativo de tomar un modelo lógico de un sistema junto con un conjunto de objetivos fuertemente establecidos para este sistema y producir las especificaciones de un sistema físico que satisfaga estos objetivos.” (Gane - Sarson). Además, es la “Actividad por la cual un relevamiento de datos y funciones de un sistema (modelo esencial) se traduce en un plan de implementación. El modelo es volcado en una tecnología determinada”. “...el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física” (E. Taylor).

El objetivo más importante del diseño es entregar las funciones requeridas por el usuario y que éstas satisfagan una especificación funcional dada.

Se considera que existe una gran brecha entre los requerimientos definidos por los usuarios de los sistemas (en conjunto con los Analistas de Software) y los diseños resultantes del análisis realizado por los Analistas de Desarrollo, por lo tanto es importante comprender que cuando se detecten de manera oportuna estas brechas, el costo de corregirlas y la implementación del desarrollo será menor; y de esta forma se impacta de manera positiva la oportunidad de la entrega y la satisfacción del cliente.

Las imágenes 4 y 5 validan esta información, reafirmando que entre más temprano se detecten las inconsistencias en el ciclo de vida del desarrollo de software, el costo de corrección será menor y el porcentaje de utilización de software después de su entrega se incrementara; se concluye que si podemos asegurar la calidad en el diseño, en las siguientes fases o etapas del desarrollo el número de errores será menor.

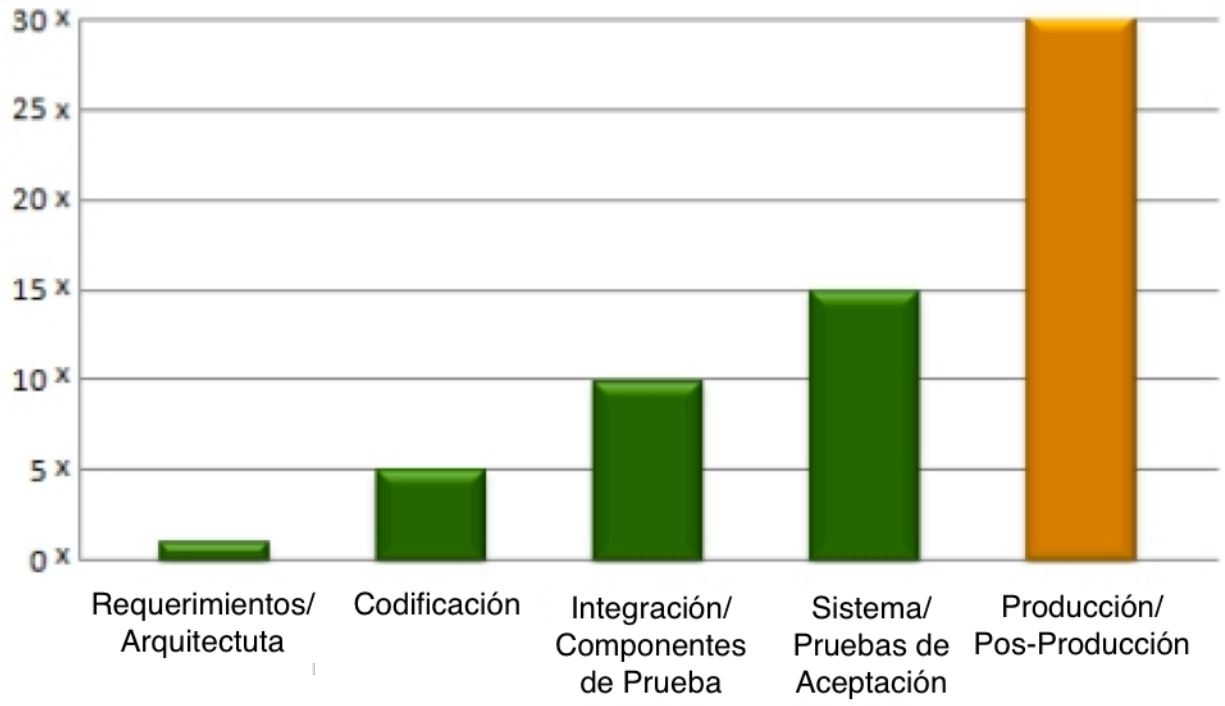

Imagen 4 Costo relativo de corrección de errores, según la etapa en que sea detectado

(Fuente: National Institute of Standards and Technology)

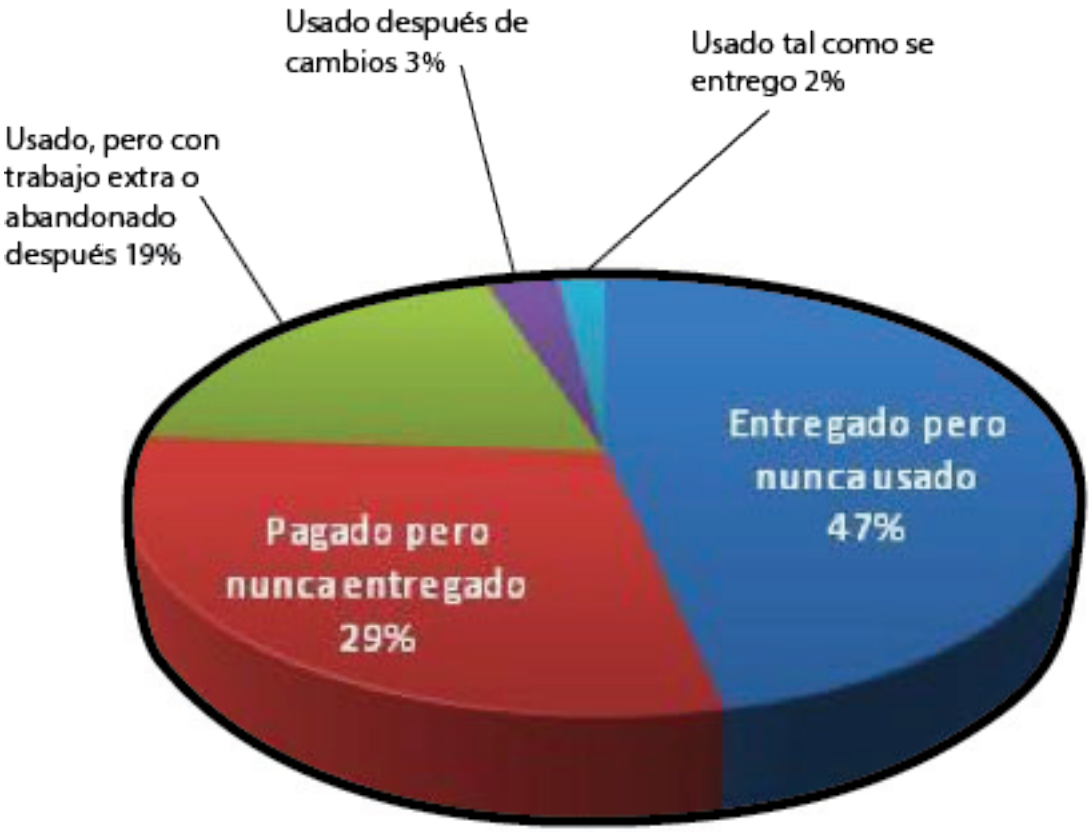

Imagen 5 Porcentaje de software utilizado después de su entrega (Fuente: Apuntes ingeniería del software de gestión. "Tema 1: Software e ingeniería del software")

El diseño de software juega un papel importante en el desarrollo de software ya que proporciona las representaciones del sistema que se puede evaluar en relación con su calidad y mejorar antes de continuar con las siguientes actividades del desarrollo, por lo anterior es que el diseño se considera el sitio en el que se establece la calidad del software.

Cabe destacar que la fase de diseño es requerida independientemente de la metodología de desarrollo de software utilizada, sin embargo sus actividades variaran de acuerdo a cada metodología.

Actualmente hay una creciente evolución en el desarrollo de software, también encontramos un notable incremento en los procesos y normas de calidad de todo el ámbito del desarrollo de software; sin embargo, ante la falta de un enfoque o una norma que regule la calidad en la fase del diseño, se ve la necesidad por medio de esta investigación, de hacer un análisis del tema, específicamente con la familia de normas ISO/IEC 25000.

Por otro lado, las tecnologías de la información cada día son más competitivas, lo cual lleva consecuentemente al crecimiento en la complejidad de los sistemas y que los clientes sean más exigentes en materia de calidad; lo anterior hace notar la importancia de implementar y definir una metodología que permita asegurar la calidad en el diseño del software.

3 Objetivos

3.1 General

Definir un modelo de evaluación de la calidad para la fase o etapa de diseño de software, basándose en la familia de estándares ISO/IEC 25000, para contribuir con la reducción de la brecha entre la etapa de requisitos y la etapa de diseño del software.

3.2 Específicos

- Analizar los conceptos y componentes de la familia de estándares ISO/IEC 25000.
- Analizar los conceptos de la normas ISO/IEC 9126, ISO/IEC 14598.
- Identifica las característica y subcaracterística de la división del modelo de calidad del producto software inmersas en la familia de normas ISO/IEC 25000.
- Identifica las métricas para la medición de la calidad producto software inmersas en la familia de normas ISO/IEC 25000.
- Levantar un modelo de aseguramiento de la calidad en el diseño software basado en la calidad en el producto de software y las métricas para la medición de la calidad
- Ejemplificar modelo propuesto (El alcance de este proyecto no contempla realizar una validación del modelo).

4 Justificación

La obtención de un Software con Calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software, que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, facilidad de mantenimiento y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de Desarrollo como para el Control de Calidad del Software.

La calidad del producto de software abarca los siguientes aspectos:

- Calidad Interna: medible a partir de las características intrínsecas, como el código fuente
- Calidad Externa: medible en el comportamiento del producto, como en una prueba
- Calidad en Uso: durante la utilización efectiva por parte del usuario

Las ventajas de implantar Modelos o Estándares de Calidad del Software, son:

- Tener una oportunidad para corregir los procesos de software que se hayan desajustado con el tiempo.
- Clasificar a las empresas como de clase mundial.
- Certificar la competitividad internacional requerida para competir en todos los mercados.
- Cambiar la actitud del personal de la empresa.
- Desarrollar y mejorar el nivel y la calidad de vida del personal.
- Generar una cultura organizacional enfocada a cumplir con los requisitos de los clientes.
- Realizar una mejora continua en la calidad de los procesos de software utilizados, servicios y productos de software.
- Lograr que la empresa de software sea más competitiva.
- Reducir los costos en todos los procesos.
- Aumentar la productividad, efectividad y utilidad de la empresa.

- Asegurar la satisfacción de los clientes internos y externos.
- Tener productos de software y servicios con valor agregado.
- Tener aceptación total de los clientes.
- Tener permanentemente mejores procesos, productos de software y servicios.
- Tener criterios de medición e indicadores congruentes que se utilizan en la empresa para comparar respecto de las mejores prácticas, para conocer fortalezas y debilidades de la empresa; y establecer las estrategias necesarias para realizar mejoras.

El diseño deberá implementar los requisitos del modelo de Análisis, y deberá ajustarse a los requisitos implícitos que desea el cliente. Adicionalmente deberá ser una guía legible y comprensible para aquellos que generan código y para quienes comprueban y consecuentemente, dan soporte al sistema; el diseño deberá proporcionar una idea completa del sistema, enfrentándose a los dominios del comportamiento, funcionales y de datos desde la perspectiva de implementación.

Es importante comprender que un sistema es cambiante, así como el entorno que lo rodea, en este orden de ideas el diseño también deberá tener la capacidad de cambiar cuando este se esté definiendo, una vez definido el diseño cambios que se deseen aplicar deberán ser controlados.

Como principios básicos del diseño tenemos que, éste deberá tener en cuenta enfoques alternativos; deberá poderse rastrear hasta el modelo de Análisis, deberá minimizar la distancia intelectual entre el sistema y el problema, presentando uniformidad e integración, se deberá estructurar de manera tal que admita cambios y deberá ser evaluado en función de la calidad.

Este trabajo de grado se centra en la necesidad de investigar un modelo o estándar que se pueda aplicar a la etapa del diseño del software, que refleje resultados significativos. Actualmente existen varios métodos y estándares internacionales que se aplican al proceso de evaluación y selección del software, enfocándose principalmente en la adquisición y el ciclo de desarrollo de los productos y servicios mediante un proceso de calidad. Muchos de estos se encuentran soportados por las normas desarrolladas y publicadas por la International Organization for Standarization (ISO), los cuales tienen gran aceptación por la comunidad científica mundial.
La evaluación de la calidad de un producto se logra mediante la definición adecuada de especificaciones y de la observación del comportamiento de dicho producto. En la ISO/IEC 25000 y la ISO/IEC 9126 se dan las pautas necesarias para el diseño e implementación de un modelo de calidad que entregue resultados con base en las características de calidad definidas, de acuerdo con necesidades. (ISO, 2006).
Se mencionan algunos estándares y modelos.
Algunos estándares son:

- ISO/IEC 9126.
- ISO/IEC 25000.

Algunos modelos son:

- Métricas (Programa) proyecto de grado universidad EAFIT.
- A model for software product quality.
- MOSCA.
- COSTUME: Un método para la combinación de modelos de calidad

Hacia la medición de calidad en uso web.

Es un reto importante para la industria del software implementar un modelo o estándar de calidad indicado, y para este trabajo de investigación en específico, la aplicación de estos estándares como el ISO/IEC 25000 en la etapa del diseño del software; permitiendo enfrentar en el mercado mayores posibilidades de éxito y que se abra el camino para un alto interés y compromiso hacia la incorporación de este estándar hacia la calidad en el diseño del software.

Aplicar un estándar de calidad a la etapa del diseño ayuda a disminuir las fallas y defectos, lo que asegura mayor calidad y menores costos a largo plazo y también más beneficios para la fase de mantenimiento del software (ISO, 2010).

Como ya se ha mencionado este trabajo plantea el Estándar de Calidad del Software a nivel del producto ISO 25000 (SQUARE). Los beneficios de utilizar SQuaRE son:

a) El modelo representa la calidad esperada del producto de software.
b) Planteo del desdoblamiento de las necesidades o expectativas en calidad en uso, calidad externa y calidad interna.
c) Permite una mayor eficacia en la definición del software.
d) Plantea la evaluación de productos intermedios.
e) Propone una calidad final a través de las evaluaciones intermedias.
f) Permite efectuar un rastreo entre las expectativas, requisitos y medidas de evaluación.
g) Mejora la calidad del producto.

5 Marco teórico

5.1 Qué es calidad?

Para definir calidad encontramos un conjunto de definiciones que pueden ayudar a entender este concepto.

A continuación se mencionan alguna de ellas.

- Conjunto de propiedades inherentes a un objeto que permiten apreciarlo como mejor, igual o peor que otros objetos de su especie. (DARE: Diccionario de la Real Academia de la Lengua).
- Conjunto de propiedades y características de un producto o servicio que le confieren capacidad para satisfacer necesidades expresadas o implícitas. (ISO 8042:1994).
- Grado en el que un conjunto de características inherentes cumplen con los requisitos. (ISO 9000:2000).

5.2 Qué es calidad de software?

Por otro lado encontramos algunas definiciones para calidad de software, que dan una idea de este concepto.

- “La calidad del Software es el grado con el que el sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (IEEE, Standard 610-1990).
- “Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario”. (Pressman, 2005).
- “La totalidad de las características de una entidad que influyen en su aptitud para satisfacer las necesidades establecidas e implícitas”. (ISO 8402).

5.3 Atributos de la calidad del software

El estándar ISO/IEC 9126 distingue entre calidad interna y calidad externa, e introduce también el concepto de calidad en uso.

- **Calidad interna:** Tiene como objetivo medir la calidad del software mediante factores medibles durante su desarrollo.

  Los factores internos hacen referencia a las características constructivas de los componentes, que son tan solo accesibles y controlables por sus fabricantes.
- **Calidad externa:** Pretende medir la calidad del software teniendo en cuenta el comportamiento de este software en un sistema del cual forme parte.

  Los factores externos son todos aquellos factores que pueden ser directamente percibidos por los usuarios y que afectan su trabajo (usualmente relacionadas a la funcionalidad y usabilidad).
- **Calidad en uso:** Corresponde a la calidad del software desde el punto de vista de un usuario.

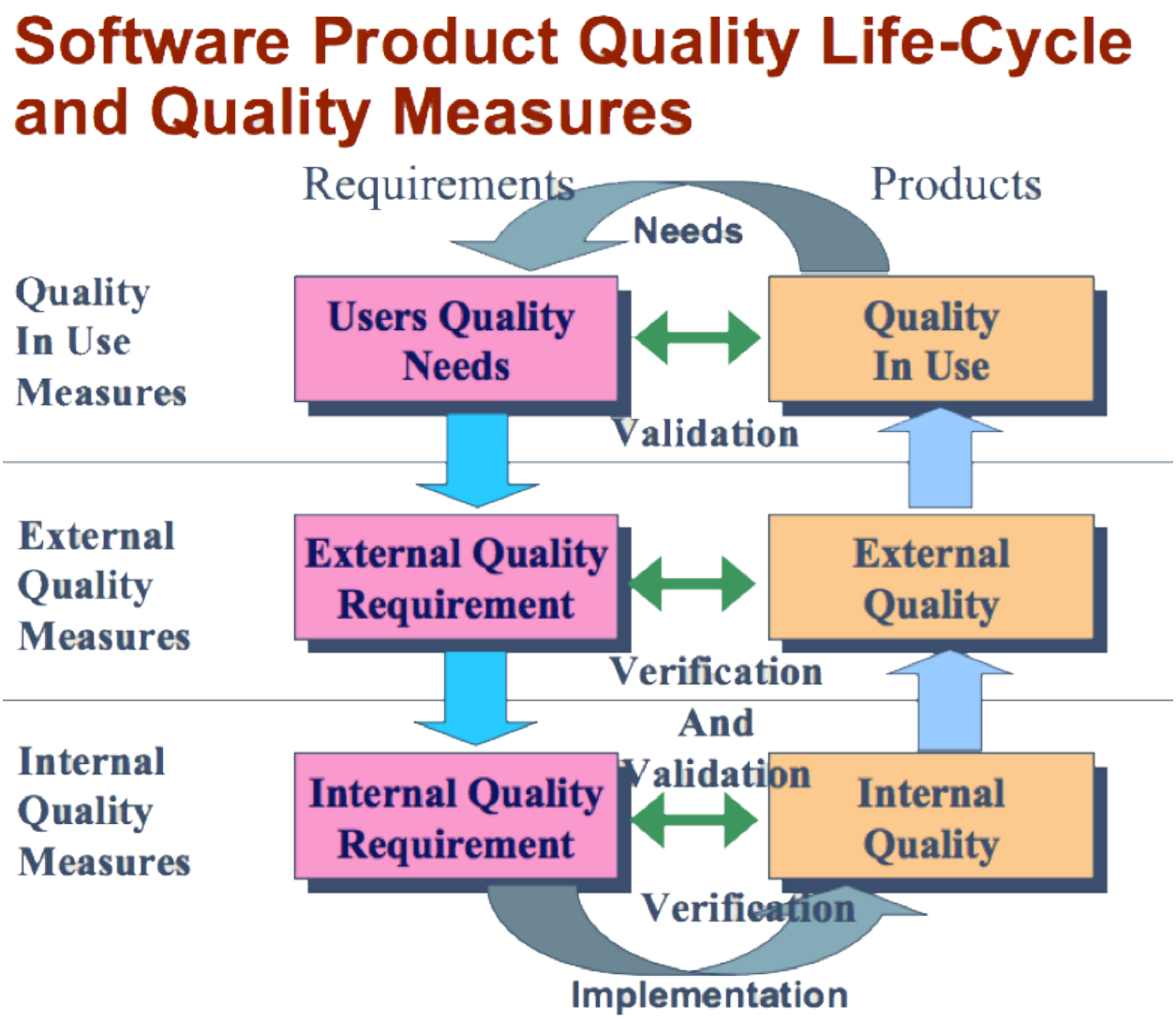

Imagen 6 Ciclo de vida de calidad del software (Fuente: ISO/IEC 9126).

5.3.1 Calidad interna y externa

- **Funcionalidad:** La capacidad del producto software para proporcionar funciones declaradas e implícitas cuando se usa bajo condiciones especificadas.
    - Adecuación: La capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuarios especificados.

  - Exactitud: La capacidad del producto software para proporcionar los resultados o efectos correctos o acordados con el grado necesario de precisión.
  - Interoperabilidad: La capacidad del producto software para interactuar con uno o más sistemas especificados.
  - Seguridad de acceso: La capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.
  - Cumplimiento funcional: La capacidad del producto software para adherirse a normas, convenciones o legislación y prescripciones similares relacionadas con la funcionalidad.
- **Fiabilidad:** La capacidad del producto software para mantener un nivel especificado de prestaciones cuando se usa bajo condiciones especificadas.
  - Madurez: La capacidad del producto software para evitar fallas como resultado de fallas en el software.
  - Tolerancia a fallos: La capacidad del producto software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificadas.
  - Capacidad de recuperación: La capacidad del producto software para restablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.
  - Cumplimiento de la fiabilidad: La capacidad del producto software para adherirse a normas, convenciones o legislación relacionadas con la fiabilidad.

- **Usabilidad:** La capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas.
    - Capacidad para ser entendido: La capacidad del producto software que permite al usuario entender si el software es adecuado y como puede ser usado para unas tareas o condiciones de uso particular.
    - Capacidad para ser aprendido: La capacidad del producto software que permite al usuario aprender sobre su aplicación.
    - Capacidad para ser operado: La capacidad del producto software que permite al usuario operarlo y controlarlo.
    - Capacidad de atracción: La capacidad del producto software para ser atractivo al usuario.
    - Cumplimiento de la usabilidad: La capacidad del producto software para adherirse a normas, convenciones, guías de estilo o legislación relacionadas con la usabilidad.
- **Eficiencia:** La capacidad del producto software para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.
    - Comportamiento temporal: La capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y rendimiento apropiados bajo condiciones determinadas.
    - Utilización de recursos: La capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.

  - Cumplimiento de la eficiencia: La capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

- **Mantenibilidad:** La capacidad del producto software para ser modificado.

  Las modificaciones podrían incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y requisitos y especificaciones funcionales.

  - Capacidad para ser analizado: La capacidad del producto software para ser diagnosticado, identificando deficiencias o causas en los fallos del software, o para identificar las partes que han de ser modificadas.
  - Capacidad para ser cambiado: La capacidad del producto software que permite que una determinada modificación sea implementada.
  - Estabilidad: La capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.
  - Capacidad para ser probado: La capacidad del producto software que permite que el software modificado sea validado.
  - Cumplimiento de la mantenibilidad: La capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

- **Portabilidad:** La capacidad del producto software para ser transferido de un entorno a otro.

  - Adaptabilidad: La capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para ese propósito por el propio software considerado.
  - Instalabilidad: La capacidad del producto software para instalado en un entorno especificado.

- Coexistencia: La capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.
- Capacidad para reemplazar: La capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.
- Cumplimiento de la portabilidad: La capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad.

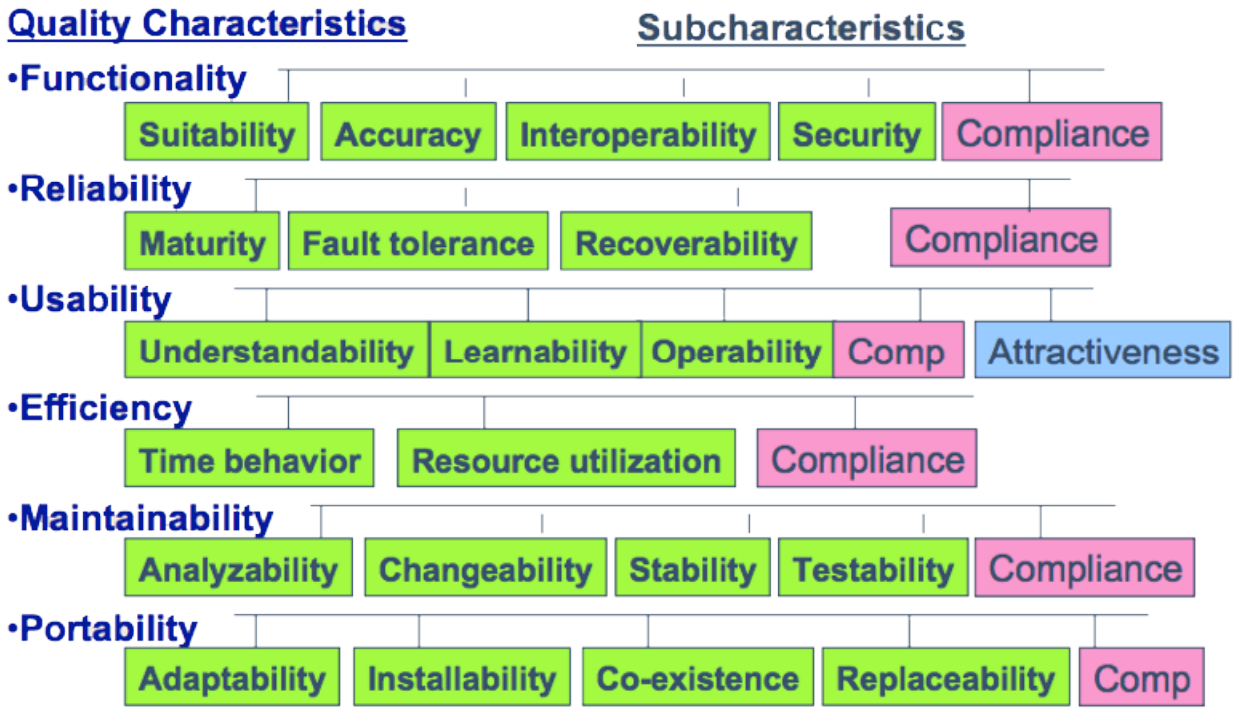

Imagen 7 Modelo de calidad ISO/IEC 9126-1 (Fuente: ISO/IEC 9126).

5.3.2 Calidad de uso

- **Efectividad:** La capacidad del producto software para permitir a los usuarios alcanzar objetivos especificados con exactitud y completitud, en un contexto de uso especificado.
- **Productividad:** La capacidad del producto software para permitir a los usuarios gastar una cantidad adecuada de recursos con respecto a la efectividad alcanzada, en un contexto de uso especificado.
- **Seguridad de acceso:** La capacidad del producto software para alcanzar niveles aceptables del riesgo de hacer daño a personas, al negocio, a las propiedades o al medio ambiente, en un contexto de uso adecuado.
- **Satisfacción:** La capacidad del producto software para satisfacer a los usuarios en un contexto de uso especificado.

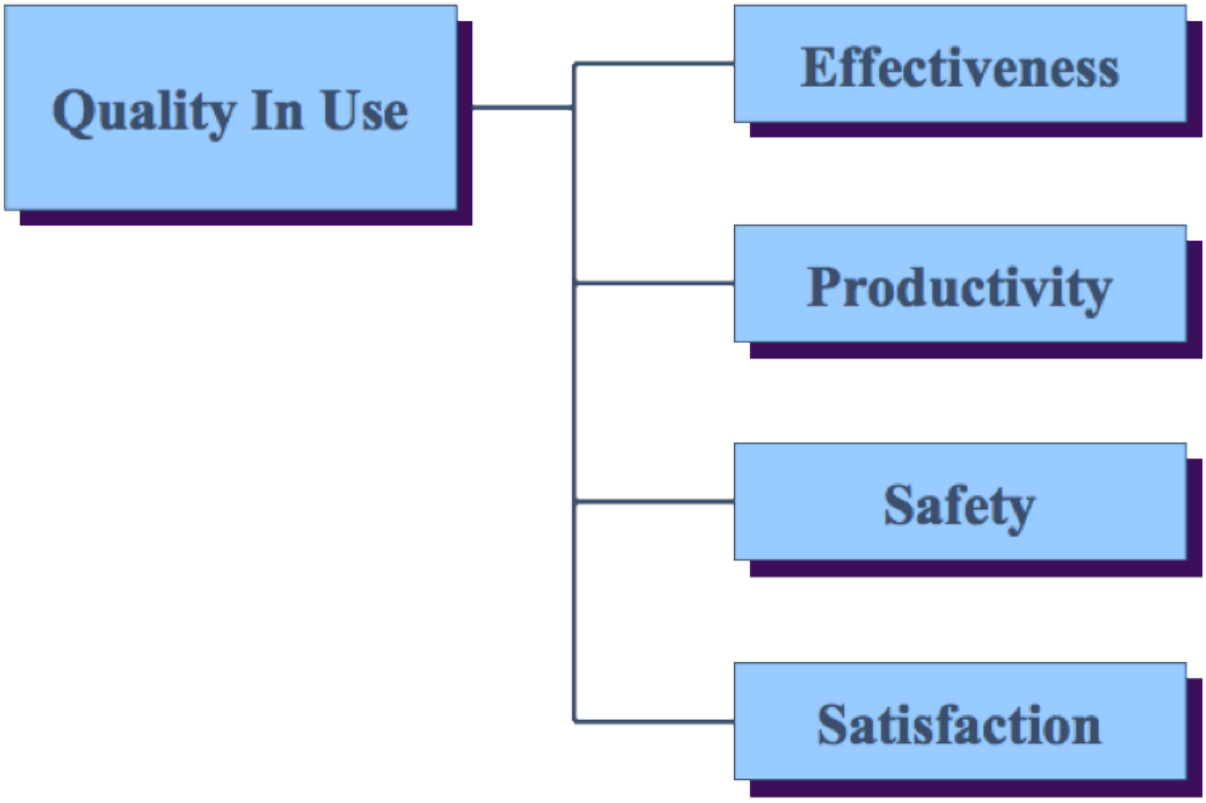

Imagen 8 Calidad en uso (Fuente: ISO/IEC 9126).

5.4 Gestión de calidad

Se han publicado varios estándares relacionados con la calidad en general, y en particular, con calidad en el software, por parte de la Organización Internacional de Estándares (ISO).

Estándares como ISO/IEC 8402, ISO/IEC 9000, ISO/IEC 9126, ISO/IEC14598 definen calidad del software como la capacidad de un conjunto de características de un producto, sistema o proceso para satisfacer requisitos de clientes y otras partes interesadas.

La gestión de la calidad del software dentro de este contexto es definida como todas las acciones coordinadas para dirigir y controlar una organización con respecto a la calidad del software.

5.5 Procesos del ciclo de vida

El ciclo de vida de un software es el período de tiempo que comienza con la concepción de la idea de un software y que termina con la vida útil del mismo. Durante este período de tiempo cooperan un conjunto de procesos interrelacionados, denominados procesos del ciclo de vida, con el objetivo de construir un producto de software de calidad. Los modelos y estándares internacionales como ISO, IEEE, identifican procesos que componen el ciclo de vida de un software. En estos estándares se identifican unas áreas de procesos:

- Procesos primarios de ingeniería: son las actividades primarias del ciclo de vida, aquellas incluidas en las disciplinas técnicas. Independientemente del modelo de ciclo de vida seleccionado (ej. Cascada, Espiral, V, W), siempre será necesario el Análisis de requisitos, Diseño, Implementación, Validación Y Verificación, Y Mantenimiento.
- Procesos de gestión de proyectos: cubre las actividades de estimación, planificación del proyecto y asignación de recursos, medición del progreso, seguimiento y control del proyecto, gestión de riesgos y gestión de las relaciones con los clientes.
- Procesos de aseguramiento de la calidad: son actividades sistemáticas y planificadas, necesarias para dirigir y controlar los procesos del ciclo de vida con el objetivo de proporcionar suficiente confianza de que el proceso y los productos del desarrollo satisfacen aceptablemente estándares de calidad. Estas actividades ejercen, por tanto, una función de watchdog, controlando todos los procesos del ciclo de vida de software.

La siguiente tabla ilustra la organización de procesos en las diferentes áreas:

	ÁREAS DE PROCESOS		
PROCESOS	Ingeniería	Gestión	Calidad
	análisis de requisitos	estimación	prevención
	diseño	planificación	detección y corrección
	implementación	medición	evaluación y mejora
	validación y verificación	control y seguimiento	
	mantenimiento	gestión de riesgos	
		relaciones con clientes	

Imagen 9: Organización de procesos en las diferentes áreas. (Autor, Ramón Mollineda, Tanja Vos, calidad y testeo del software)

En este trabajo se hace énfasis en la etapa del Diseño y cómo se pueden aplicar estándares de calidad en esta etapa.

5.6 Diseño de software

El diseño es el primer paso de la fase de desarrollo de cualquier producto de sistema de ingeniería.

“Diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso, o sistema, con los suficientes detalles como para permitir su realización física” (E.S.Taylor, An Interim Report on Engineering Design, Massachusetts Institute of Technology, 1959)

El Diseño es el proceso de definición de la arquitectura del sistema, de las estructuras de datos y de los algoritmos a emplear, antes de realizar la construcción del software. Algunos fundamentos

que garantizan diseños robustos son el conocimiento de estilos (estructurado, OO) y conceptos (modularidad, abstracción) básicos de diseño, algoritmos y estructuras de datos primarias, esquemas típicos de arquitecturas, herramientas de diseño, entre otros.

El objetivo del diseñador es producir un modelo de una entidad que se construirá más adelante. El proceso por el cual se desarrolla el modelo combina:

- La intuición y los criterios en base a la experiencia de construir entidades similares.
- Un conjunto de principios y/o heurísticas que guían la forma en la que se desarrolla el modelo.
- Un conjunto de criterios que permiten discernir sobre calidad.
- Un proceso de iteración que conduce finalmente a una representación del diseño final.

Los ciclos de vida modernos de software prestan especial atención al diseño de arquitectura, cuya solución suele ser una tarea prioritaria. Organizaciones preocupadas por la calidad de su proceso de software documentan soluciones genéricas de diseño en función del dominio de aplicación a resolver, e incluyen experiencias previas de la aplicación de estas soluciones.
La idea de realizar diseño de software en lugar de “programar”, surgió a principios de los años 60, por lo que a las metodologías de diseño les falta la profundidad y la flexibilidad que tiene el diseño en otras ingenierías.

5.6.1 Ingeniería de software y diseño de software

Una vez que se han establecido los requisitos del software, el diseño es la primera de tres actividades técnicas: diseño, codificación y prueba. Cada actividad transforma la información de forma que al final se obtiene un software validado.

El diseño es técnicamente la parte central de la ingeniería del software. Durante el diseño se desarrollan, revisan y se documentan los refinamientos progresivos de las estructuras de datos, de la estructura del programa y de los detalles procedimentales. El diseño da como resultado representaciones cuya calidad puede ser evaluada.

Mediante algunas metodologías de diseño se realiza el diseño de datos, el diseño arquitectónico y el diseño procedimental.

- El **diseño de datos** transforma el modelo de campo de información, creado durante el análisis, en las estructuras de datos que se van a requerir para implementar el software.
- El **diseño arquitectónico** define las relaciones entre los principales elementos estructurales del programa.
- El **diseño procedimental** transforma los elementos estructurales en una descripción procedimental del software. El diseño procedimental se realiza después de que se ha establecido la estructura del programa y de los datos. Define los algoritmos de procesamiento necesarios. Concluido el diseño se genera el código fuente y para integrar y validar el software, se llevan a cabo pruebas de testeo.

Las fases de diseño, codificación y prueba absorben el 75% o más del coste de la ingeniería del software (excluyendo el mantenimiento). Es aquí donde se toman las decisiones que afectarán finalmente al éxito de la implementación del programa, y también, a la facilidad de mantenimiento que tendrá el software. Por tanto el diseño es un paso fundamental de la fase de desarrollo.

El diseño es la forma mediante la cual se puede traducir los requerimientos del cliente, es la base de todas las partes posteriores del desarrollo y de la fase de prueba del software.

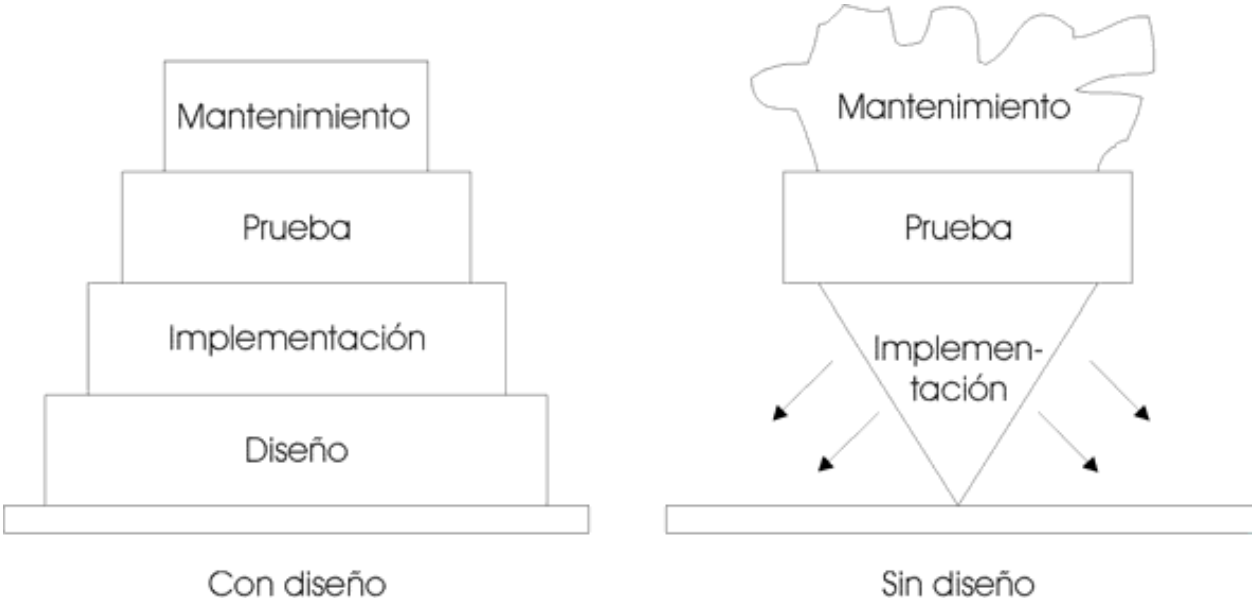

Imagen 10: Importancia del diseño. (Fuente: Pressman, 2005)

Sin diseño, nos arriesgamos a construir un sistema inestable, un sistema que falle cuando se realicen pequeños cambios, un sistema que sea difícil de probar, un sistema cuya calidad no pueda ser evaluada hasta más adelante, cuando quede poco tiempo y se haya gastado mucho dinero.

5.6.2 El proceso de diseño

El diseño del software es un proceso mediante el cual se traducen los requisitos en una representación del software, que se acerca mucho al código fuente.

Desde el punto de vista de la gestión del proyecto, el diseño del software se realiza en dos etapas: el diseño preliminar y el diseño detallado.

- El diseño preliminar se centra en la transformación de los requisitos en los datos y la arquitectura del software.
- El diseño detallado se ocupa del refinamiento y de la representación arquitectónica que lleva a una estructura de datos refinada y a las representaciones algorítmicas del software.

Además del diseño de datos, del diseño arquitectónico y del desarrollo procedimental, muchas aplicaciones modernas requieren un diseño de la interfaz.

5.6.3 Diseño y calidad del software

Durante el proceso de diseño, la calidad se puede evaluar mediante una serie de revisiones técnicas formales (RTF) que son un filtro para el proceso de ingeniería de software y se aplican en varios momentos del desarrollo, por lo tanto, estas sirven para detectar fallos tanto en el análisis como en el diseño y la codificación de manera que puedan ser eliminados cuanto antes. Los objetivos de las revisiones técnicas formales que son aplicables a la etapa del diseño son:

- Verificar que el software alcanza sus requisitos.
- Garantizar que el software se ha representado según los estándares establecidos.
- Conseguir un software desarrollado de forma uniforme.
- Hacer que los proyectos sean manejables.

Criterios para determinar la calidad del software:

- Un diseño debe tener una organización jerárquica.

- Un diseño debe ser modular, es decir, el software debe estar dividido en elementos que realicen funciones específicas.
- Un diseño debe tener representaciones distintas y separadas de los datos y de los procedimientos.
- Un diseño debe llevar a módulos que exhiban características funcionales independientes.
- Un diseño debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el exterior.
- Un diseño debe obtenerse mediante un método que sea reproducible y que esté dirigido por la información obtenida durante el análisis de requerimientos.

Para un buen diseño de software se debe tener una metodología sistemática y una revisión exhaustiva.

5.7 Modelos ISO para la calidad del producto software

5.7.1 Organización de la ISO e IEC

La ISO (International Standards Organization) es una organización de estandarización en cuyo funcionamiento intervienen organismos de todo el mundo interesados en regular y armonizar diversas áreas de la industria. Por su propia naturaleza, la ISO emite normas conocidas como “de facto”, es decir, normativas cuya adopción no es obligatoria legalmente.

Sin embargo, los beneficios de la estandarización hacen que los documentos emitidos por la ISO sean adoptados rápidamente o al menos tenidos en cuenta a la hora de diseñar, implantar o mejorar un producto o servicio.

La ISO es una organización no gubernamental que actúa como puente entre la empresa pública y la privada y que según sus propias palabras “permite ofrecer al público productos con

características atractivas o interesantes como el respeto al medio ambiente, eficiencia o productos con piezas intercambiables entre distintos fabricantes".

La IEC (International Electrotechnical Comission) es una organización que también trabaja a nivel mundial en el desarrollo de estándares sobre electricidad y electrónica. Según sus propios estatutos, la IEC "tiene como objetivo promover la normalización en todas las cuestiones relacionadas con la electrotécnica".

Tanto en la ISO como en la IEC hay diversos comités técnicos (Technical Comittees) en los que intervienen los organismos de cada nación y que participan en el desarrollo de estándares en un determinado ámbito. En ocasiones, la ISO y la IEC colaboran en los llamados Comités Técnicos Conjuntos (Joint Technical Committees). Uno de estos comités técnicos es el llamado ISO/IEC JTC 1, especializado en las tecnologías de la información. A menudo, un JTC puede estar dividido en varios subcomités (SubCommittees) que son los que a menudo elaboran el grueso de un estándar y que dan nombre a los estándares. En este JTC 1 de ISO/IEC hay un SC con el número 7, responsable del área de Ingeniería del Software y de Sistemas.

Este trabajo versa sobre las normas ISO/IEC JTC 1/SC 7 de la serie 25000 denominadas "Software Product Quality Requirements and Evaluation" y conocidas más comúnmente con el nombre "SQuaRE". (ISO/IEC 25000:2005).

5.7.2 El estándar ISO/IEC 9126

El estándar ISO/IEC 9126 (2001), presenta un marco conceptual para el modelo de calidad y define un conjunto de características, refinadas en subcaracterísticas, las cuales debe cumplir todo producto software para ser considerado de calidad.

En este estándar, se define un modelo de calidad como “El conjunto de características y las relaciones entre las mismas, que proveen la base para especificar requerimientos de calidad y evaluar calidad”.

En relación con el modelo de calidad del producto software, el estándar ISO/IEC 9126 (2001), está dividido en cuatro partes:

- **ISO/IEC 9126-1 (2001):** Presenta un modelo de calidad del software, estructurado en características y subcaracterísticas.
- **ISO/IEC 9126-2 (2003):** Proporciona métricas externas para medir los atributos de las seis características de calidad externa definidas en la ISO/IEC 9126-1 (2001) y una explicación de cómo aplicar las métricas de calidad de software.
- **ISO/IEC 9126-3 (2003):** Proporciona métricas internas para medir atributos de seis características de calidad interna definidas en la ISO/IEC 9126-1 (2001).
- **ISO/IEC 9126-4 [39]:** Define métricas de calidad en uso para medir los atributos definidos en la ISO/IEC 9126-1 (2001).

El estándar ISO9126-1(2001), presenta dos modelos de calidad. La primera referida a la calidad interna y externa y la segunda a la calidad en uso. A continuación se definen las características descriptas en la ISO/IEC 9126 (2001) y citadas en Abrahão et al. (2001):

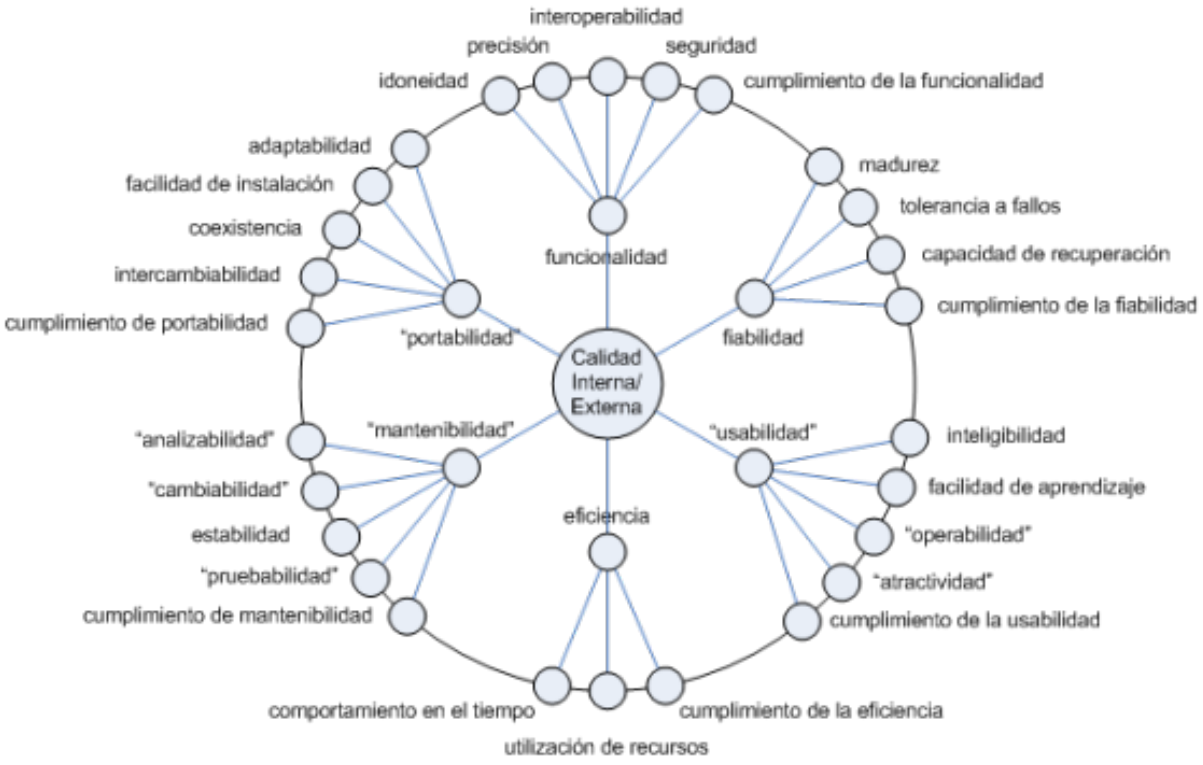

Imagen 11 Características de la calidad interna y externa según la ISO/IEC 9126. (Fuente: ISO/IEC 25000:2005).

5.7.2.1 Funcionalidad

Conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- **Idoneidad:** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.
- **Precisión:** Capacidad del producto software para proporcionar los resultados o efectos correctos acordados, con el grado necesario de precisión.

- **Interoperabilidad:** Capacidad del producto software para interactuar con uno o más sistemas especificados.
- **Seguridad:** Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.
- **Cumplimiento funcional:** Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

5.7.2.2 Fiabilidad

Conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

- **Madurez:** Capacidad del producto software para evitar fallar como resultado de fallos en el software.
- **Tolerancia a fallos:** Capacidad del software para mantener un nivel especificado de prestaciones en cado de fallos de software o de infringir sus interfaces especificados.
- **Capacidad de recuperación:** Capacidad del producto software para restablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.
- **Cumplimiento de la fiabilidad:** Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con la fiabilidad.

5.7.2.3 Usabilidad

Conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

- **Capacidad para ser entendido:** Capacidad del producto software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares.
- **Capacidad para ser aprendido:** Capacidad del producto software que permite al usuario aprender sobre su aplicación.
- **Capacidad para ser operado:** Capacidad del producto software que permite al usuario operarlo y controlarlo.
- **Capacidad de atracción:** Capacidad del producto software para ser atractivo para el usuario.
- **Inteligibilidad:** Capacidad del producto software que permite al usuario entender si el software es adecuado y como puede ser usado para unas tareas o condiciones de uso particular.
- **Cumplimiento de la usabilidad:** Capacidad del producto software para adherirse a normas, convenciones, guías de estilos o regulaciones relacionadas con la usabilidad.

5.7.2.4 Eficiencia

Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- **Comportamiento en el tiempo:** Capacidad del producto software para proporcionar tiempos de respuestas, tiempos de proceso y potencia apropiados, bajo condiciones determinadas.
- **Utilización de recursos:** Capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Cumplimiento de eficiencia:** Capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

5.7.2.5 Mantenibilidad

Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- **Capacidad para ser analizado::** Es la capacidad del producto software para serle diagnosticado deficiencias o causas de fallos en el software o para identificar las partes que han de ser modificadas.
- **Capacidad para ser cambiado:** Capacidad del producto software que permite una determinada modificación sea implementada.
- **Estabilidad:** Capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.
- **Capacidad para ser probado:** Capacidad del producto software que permite que el software sea validado.
- **Cumplimiento de mantenibilidad:** Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

5.7.2.6 Portabilidad

Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

- **Facilidad de instalación:** Capacidad del producto software para ser instalado en un entorno especificado.
- **Capacidad para reemplazar:** Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.
- **Adaptabilidad:** Capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.
- **Co-existencia:** Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.
- **Cumplimiento de portabilidad:** Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad.

5.7.3 Calidad interna y externa

En Olsina et al. (2005) citado en Covella (2005), se sintetizan los enfoques de calidad interna y externa del producto software, en el estándar ISO 9126-1(2001).

- **Calidad interna:** Especificada por un modelo de calidad similar al modelo ISO/IEC 9126. Puede ser medida y evaluada por medio de atributos estáticos de documentos tales como: i) Especificación de requerimientos, ii) Arquitectura o diseño, iii) Piezas de código fuente, entre otros. En etapas tempranas del ciclo de

vida del software es posible medir, evaluar y controlar la calidad interna de estos productos. Sin embargo, asegurar la calidad interna no es generalmente suficiente para asegurar la calidad externa.

- **Calidad externa:** Especificada también por un modelo de calidad similar al modelo ISO/IEC 9126. Puede ser medida y evaluada por medio de propiedades dinámicas del código ejecutable en un sistema de computación, esto es, cuando un módulo o la aplicación completa es ejecutado en una computadora o en una red simulando lo más cercanamente posible un ambiente real. En fases tardías del ciclo de vida del software (principalmente en distintas etapas de testing o ya en estado operativo de un producto de software o aplicación Web), es posible medir, evaluar y controlar la calidad externa de estos productos ejecutables.

La calidad interna expuesta ISO/IEC 9126-1(2001), se define como "la totalidad de atributos de un producto que determina su capacidad de satisfacer necesidades explícitas e implícitas cuando es usadas bajo condiciones específicas". Se define como calidad externa "el grado en la que un producto satisface necesidades explícitas e implícitas cuando se utiliza bajo condiciones especificadas" (Covella, 2005).

Para los modelos de calidad interna y externa, se mantuvieron en la revisión las seis características principales de calidad. Aún más, a nivel de subcaracterísticas se transformaron en prescriptivas en vez de informativas. Además, se añadieron nuevas subcaracterísticas y otras redefinidas en términos de "capacidad del software" para facilitar la interpretación de las mismas desde una perspectiva de calidad interna o de calidad externa (Covella, 2005).

- **Calidad en uso:** Respecto a la calidad en uso se menciona al estándar ISO/IEC 9126-4, que contiene ejemplos de métricas para medir la productividad, efectividad, seguridad y satisfacción.

  El estándar ISO9126-1 (2001), define calidad en uso como "la capacidad de un producto de software de facilitar a usuarios específicos alcanzar metas específicas con eficacia, productividad, seguridad y satisfacción en un contexto específico de uso". Además agrega que "calidad en uso es la visión de calidad de los usuarios de un ambiente conteniendo software, y es medida sobre los resultados de usar el software en el ambiente, antes que sobre las propiedades del software en sí mismo" (Covella, 2005).

  Las características de calidad en uso son agrupadas en cuatro categorías, expuestas en Covella (2005) y se definen como:

  - **Eficacia:** Capacidad del producto software para facilitar a los usuarios alcanzar metas específicas con exactitud y completitud en un contexto específico de uso.
  - **Productividad:** Capacidad del producto software para invertir la cantidad apropiada de recursos en relación con la eficacia alcanzada en un contexto específico de uso.
  - **Seguridad:** Capacidad del producto software para alcanzar niveles aceptables de riesgo de dañar a las personas, el negocio, el software, la propiedad o el ambiente en un contexto especifico de uso.
  - **Satisfacción:** Capacidad del producto de software para satisfacer a los usuarios en un contexto específico de uso.

La imagen 22, presenta un marco conceptual para el modelo de calidad. Se observa que la calidad del proceso, en el ciclo de vida definido en ISO/IEC 12207, contribuye a mejorar la calidad del producto y ésta a la calidad en uso. Por lo tanto, mejorar el proceso de desarrollo ayuda a obtener un producto de mejor calidad y evaluar la calidad del producto mejora la calidad en uso. El estándar ISO/IEC 14598 presenta pautas que ayudan al proceso de evaluación considerando diferentes actores (desarrolladores, evaluadores, adquisidores) (González et al. 2002). En la serie ISO/IEC 14598, se destacan las siguientes normas: i) ISO/IEC 14598-1 (1999), ii) ISO/IEC 14598-2 (2000), iii) ISO/IEC 14598-3 (2000), iv) ISO/IEC 14598-4 (1999), v) ISO/IEC 14598-5 (1998) y vi) ISO/IEC 14598-6 (2001). En la imagen 23 se muestra la relación entre ambas series.

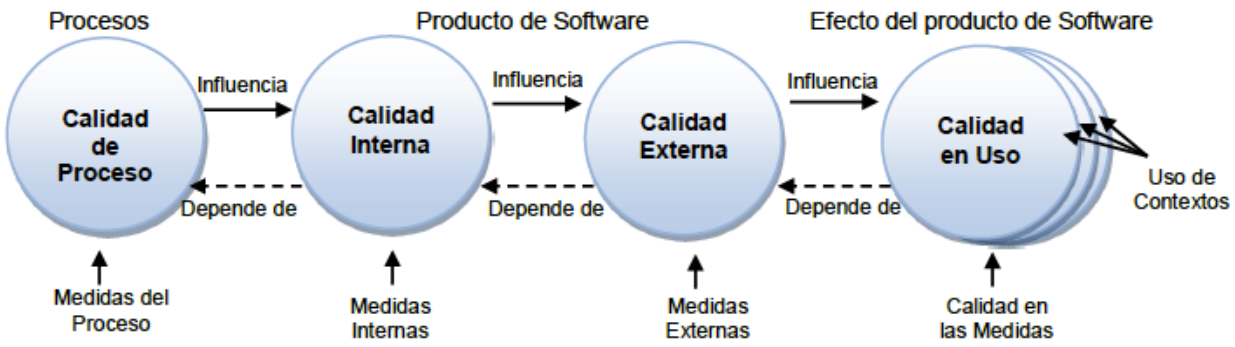

Imagen 12: Marco conceptual para el modelo de calidad (Fuente: ISO/IEC 25000:2005).

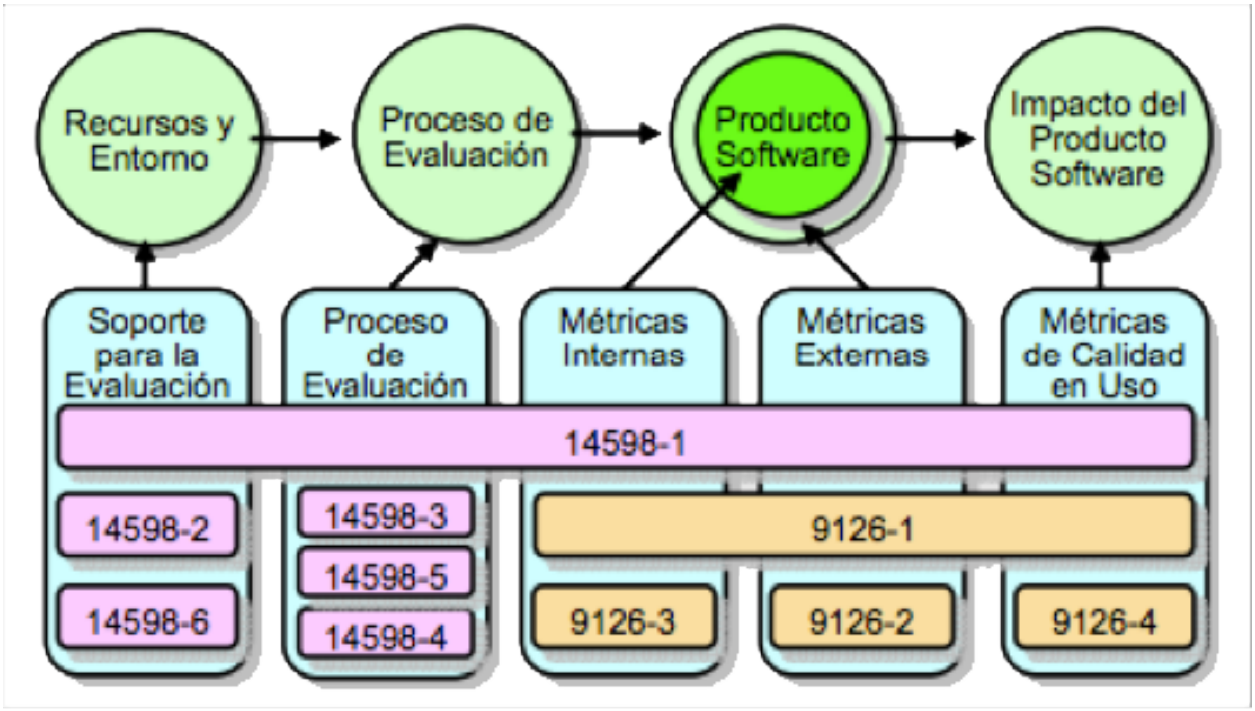

Imagen 13: Relación ISO/IEC 9126 – ISO/IEC 14598 (Fuente: ISO/IEC 9126-1, 2001).

5.7.4 Estándar ISO/IEC 25000:2005

Los aspectos más importantes en el desarrollo de software son la calidad del producto y del proceso. ISO/IEC 25000, proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE). Constituyen una serie de normas basadas en la ISO/IEC 9126 y en la ISO/IEC 14598, y su objetivo principal es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad (Portal ISO 25000).

La familia ISO/IEC 25000 está orientada al producto software, permitiendo definir el modelo de calidad y el proceso a seguir para evaluar dicho producto.

5.7.4.1 La familia de normas SQuaRE está compuesta por 5 divisiones:

5.7.4.1.1 ISO/IEC 2501n

División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.

Actualmente esta división se encuentra formada por:

- ISO/IEC 25010 - System and software quality models: describe el modelo de calidad para el producto software y para la calidad en uso. Esta Norma presenta las características y subcaracterísticas de calidad frente a las cuales evaluar el producto software.
- ISO/IEC 25012 - Data Quality model: define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información.

5.7.4.1.2 ISO/IEC 2502n

División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Presenta aplicaciones de métricas para la calidad de software interna, externa y en uso.

Actualmente esta división se encuentra formada por:

- ISO/IEC 25020 - Measurement reference model and guide: presenta una explicación introductoria y un modelo de referencia común a los elementos de medición de la calidad. También proporciona una guía para que los usuarios seleccionen o desarrollen y apliquen medidas propuestas por normas ISO.
- ISO/IEC 25021 - Quality measure elements: define y especifica un conjunto recomendado de métricas base y derivadas que puedan ser usadas a lo largo de todo el ciclo de vida del desarrollo software.
- ISO/IEC 25022 - Measurement of quality in use: define específicamente las métricas para realizar la medición de la calidad en uso del producto.
- ISO/IEC 25023 - Measurement of system and software product quality: define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.
- ISO/IEC 25024 - Measurement of data quality: define específicamente las métricas para realizar la medición de la calidad de datos.

5.7.4.1.3 ISO/IEC 2503n

División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto software que va a ser desarrollado o como entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288 (ISO, 2003).

- ISO/IEC 25030 - Quality requirements: provee de un conjunto de recomendaciones para realizar la especificación de los requisitos de calidad del producto software.

5.7.4.1.4 ISO/IEC 2504n

División de evaluación de la calidad. Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.

Esta división se encuentra formada por:

- ISO/IEC 25040 - Evaluation reference model and guide: propone un modelo de referencia general para la evaluación, que considera las entradas al proceso de evaluación, las restricciones y los recursos necesarios para obtener las correspondientes salidas.
- ISO/IEC 25041 - Evaluation guide for developers, acquirers and independent evaluators: describe los requisitos y recomendaciones para la implementación práctica de la evaluación del producto software desde el punto de vista de los desarrolladores, de los adquirentes y de los evaluadores independientes.
- ISO/IEC 25042 - Evaluation modules: define lo que la Norma considera un módulo de evaluación y la documentación, estructura y contenido que se debe utilizar a la hora de definir uno de estos módulos.
- ISO/IEC 25045 - Evaluation module for recoverability: define un módulo para la evaluación de la subcaracterística Recuperabilidad (Recoverability).

5.7.4.1.5 ISO/IEC 2500n

División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.

Actualmente esta división se encuentra formada por:

- ISO/IEC 25000 - Guide to SQuaRE: contiene el modelo de la arquitectura de SQuaRE, la terminología de la familia, un resumen de las partes, los usuarios previstos y las partes asociadas, así como los modelos de referencia.
- ISO/IEC 25001 - Planning and management: establece los requisitos y orientaciones para gestionar la evaluación y especificación de los requisitos del producto software.

Imagen 14: Divisiones ISO/IEC 25000 (Fuente: ISO/IEC 25000:2005)

La calidad del producto software forma parte central como lo menciona ISO/IEC 25000, relacionando directamente las variables de funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Según Monsalve (2010) “La calidad que pueden alcanzar los productos software, y en general cualquier producto, está sometida a como se desarrolla cada una de las etapas de la vida del producto, partiendo por la definición de la idea del producto hasta la entrega y mantención del mismo”.

ISO/IEC 25000 define tres vistas diferenciadas en el estudio de la calidad de un producto:

- **Vista interna:** Se ocupa de las propiedades del sistema como: el tamaño, la complejidad o la conformidad con las normas de orientación a objetos.

Puede utilizarse desde las primeras fases del desarrollo, permitiendo detectar deficiencias en el sistema en etapas tempranas del ciclo de vida.

- **Vista externa:** Analiza el comportamiento del sistema en producción y estudia sus atributos, por ejemplo: el rendimiento de un software en una máquina determinada, el uso de memoria de un programa o el tiempo de funcionamiento entre fallos.

  Esta vista necesita que el sistema este completo y en la fase de producción del producto, siendo muy dependiente de la máquina donde se ejecute.

- **Vista en uso:** Mide la productividad y efectividad del usuario final al utilizar el sistema.

  Por último la tercera vista que también estudia el producto software finalizado será dependiente del usuario y estará condicionada a los factores personales del mismo.

Imagen 15 Calidad del producto software (Fuente: ISO/IEC 25000:2005)

La Calidad del Software se clasifica en un conjunto estructurado de características y subcaracterísticas de la siguiente manera:

5.7.4.1.5.1 Adecuación funcional

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Completitud funcional:** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos especificados por el usuario.
- **Corrección funcional:** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Adecuación funcional:** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

5.7.4.1.5.2 Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Comportamiento temporal:** Los tiempos de respuesta y procesamiento y los ratios de rendimiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (benchmark) establecido.
- **Utilización de recursos:** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.

5.7.4.1.5.3 Compatibilidad

- **Coexistencia:** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad:** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

5.7.4.1.5.4 Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Capacidad para reconocer su adecuación:** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- **Capacidad de aprendizaje técnico:** Capacidad del producto que permite al usuario aprender su aplicación.
- **Capacidad para ser usado:** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Protección contra errores de usuario:** Capacidad del sistema para proteger a los usuarios de hacer errores.
- **Estética de la interfaz de usuario:** Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad técnica:** Capacidad del producto que permite que sea utilizado por usuarios con determinadas discapacidades.

5.7.4.1.5.5 Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Madurez:** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad:** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos:** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación:** Capacidad del producto software para recuperar los datos directamente afectados y restablecer el estado deseado del sistema en caso de interrupción o fallo.

5.7.4.1.5.6 Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Confidencialidad:** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad:** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.

- **No repudio:** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
- **Responsabilidad:** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad:** Capacidad de demostrar la identidad de un sujeto o un recurso.

5.7.4.1.5.7 Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Modularidad:** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **Reusabilidad:** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **Analizabilidad:** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado:** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado:** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

5.7.4.1.5.8 Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Adaptabilidad:** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- **Capacidad para ser instalado:** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- **Capacidad para ser reemplazado:** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Según Villalba (2009) “Los modelos de calidad son una parte fundamental en los procesos de desarrollo y evaluación de la calidad del software. El uso de estos modelos se ha generalizado sobre todo desde la aparición de modelos de calidad estándar. Estos modelos, de acuerdo a su naturaleza de estándares, constituyen modelos genéricos y no directamente aplicables a la práctica diaria, por lo que requieren de un esfuerzo adicional para adaptarlos a cada dominio de aplicación especifico. De ahí que existan multitud de trabajos en los que el objetivo es la definición de modelos de calidad reutilizables para dominios de aplicación específicos que, al no tener que ser definidos para cada proyecto desde cero, ahorren tiempo. Por otra parte, este tipo de modelos pueden ofrecer una evaluación más exacta ya que sus propiedades se pueden definir de forma más precisa.”

5.7.5 ISO/IEC 25050

- **25099**

Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software "Off -The-Shelf" y para el formato común de la industria (CIF) para informes de usabilidad.

El estándar **ISO/IEC 25000 (**2005), contiene una explicación sobre el proceso de transición entre el estándar ISO/IEC 9126, las series ISO/IEC 14598 y SQuaRE. También presenta información sobre cómo utilizar la norma ISO/IEC 9126 y la serie ISO/IEC 14598 en su forma anterior. Ofrece términos y definiciones, modelos de referencia, guía general, guías de división individual y los estándares para fines de especificación, planificación y gestión, medición y evaluación.

5.7.6 El estándar ISO/IEC 25010:2011

El estándar ISO/IEC 25010 (2011), reemplazada y actualiza el estándar ISO9126-1 (2001). Define:

- Un modelo de calidad en uso que se compone de cinco características (algunas de las cuales se subdividen en subcaracterísticas). Se relacionan con el resultado de la interacción cuando un producto se emplea en un contexto particular de uso.
- Un modelo de calidad del producto que se compone de ocho características (que se subdividen en subcaracterísticas). Se refieren a propiedades estáticas de software y las propiedades dinámicas del sistema informático. El modelo es aplicable a los productos de software y sistemas informáticos.

Las características definidas por ambos modelos son relevantes para todos los productos de software y sistemas informáticos. Las características y subcaracterísticas proporcionan coherencia terminológica para especificar, medir y evaluar la calidad del producto software y sistemas informáticos.

El modelo de calidad de producto abarca cualidades internas y externas del sistema y está compuesto por 8 características y 31 subcaracterísticas. El modelo en uso se compone de 5 características y 9 subcaracterísticas (Polillo, 2011).

La norma establece tres vistas para determinar en el estudio la calidad del producto. Primero se realiza una vista interna que se ocupa de examinar las propiedades del software. En segundo lugar se realiza una vista externa que analiza el comportamiento que tiene el software en productividad. En tercer y último lugar se hace la vista en uso que mide la efectividad del software.

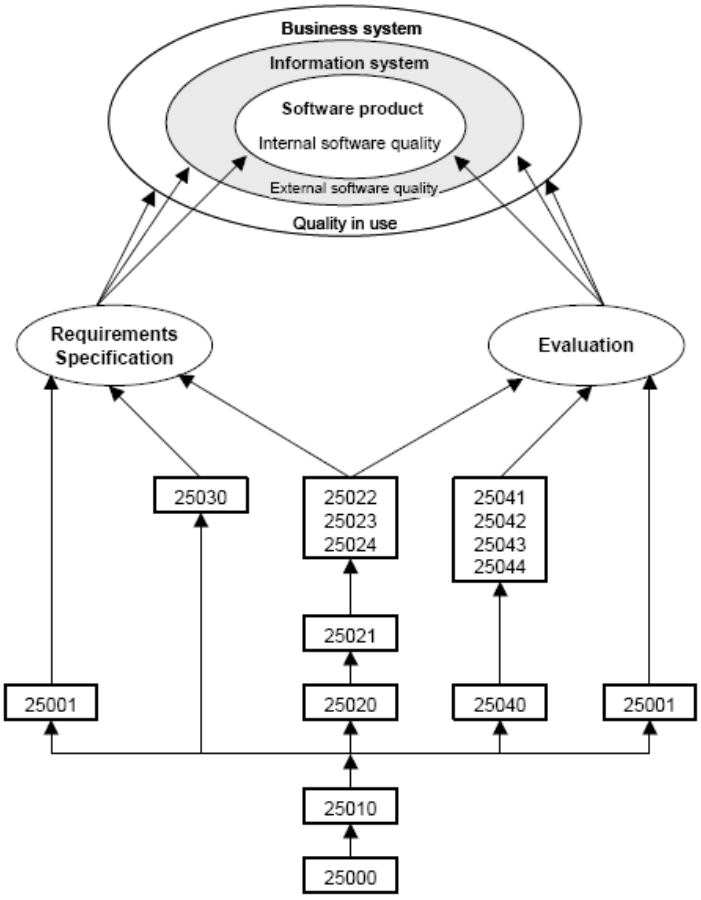

Imagen 16: El modelo de referencia de SQuaRE (Fuente: ISO/IEC 25000:2005)

5.7.7 ISO/IEC 15939

- La norma ISO/IEC15939 “define los procesos necesarios para mejorar el proceso de medida del software e identificar cómo debe ser dicho proceso para poder atender correctamente las necesidades de información técnica y de gestión del software” [4]. Los propósitos de esta norma son: Proporcionar un proceso unificado, pero adaptable, para la medida. Este proceso da soporte a las medidas específicas utilizadas en la ingeniería del software.

- Permitir que el proceso pueda mezclar datos sobre el proceso y el producto para obtener información significativa.
- Proporcionar una terminología común.
- Establecer las bases para la recogida y uso de medidas que forma que puedan utilizarse para la estimación, seguimiento y análisis de proyectos software.

En este sentido, SQuaRE proporciona todas estas características permitiendo a la vez tanto la especificación de requisitos de calidad, como la evaluación de la misma.

6 Modelo de aseguramiento de la calidad en el diseño del software

Considerando el análisis realizado en los capítulos anteriores, y de acuerdo con las características que aporta la familia de estándares de calidad ISO/IEC 25000, se hace necesario describir paso a paso la metodología para asegurar la calidad en el diseño de software de forma comprensible, de tal manera que pueda aplicar a los proyectos en la etapa de diseño.

La metodología propuesta se encuentra estimada en cuatro fases, que utiliza la metodología PHVA: Planear, Hacer, Verificar y Actuar; estas se deducen de diversos estándares de evaluación, como la gestión de calidad del proyecto y mejora continua, ISO 9000:2000.

El objetivo de tener estas 4 fases es asegurar el cumplimiento del objetivo propuesto en el menor tiempo posible.

6.1 Planear

Establecer la manera (el camino, el método) para alcanzar las metas propuestas.

Etapa inicial que tiene como finalidad establecer los aspectos con los cuales se va a asegurar la calidad del diseño a implementar.

Para el aseguramiento en la calidad se deben considerar aspectos como: el reconocimiento del entorno (tecnología), definición del alcance, determinación de objetivos, identificación de actividades, recursos necesarios, mesas de trabajo con el equipo de TI, mesas de trabajo con los stakeholders, contextualización de los requisitos elicitados, y aclaración de alcances con los usuarios.

Esta etapa se caracteriza por involucrar a la gente de todo el contexto del proyecto (stakeholders), se realizan mesas de trabajo en donde se logre determinar los alcances del sistema, adicionalmente se recopilan datos críticos del entorno, como lo son las personas que van

a utilizar el sistema y necesidad a solucionar con el sistema que se quiere implementar, esto con el fin de comprender un poco más las necesidades del usuario.

Es importante una vez realizadas las mesas de trabajo con los stakeholders, realizar mesas de trabajo con el equipo de TI, que ayudarán a darle un orden a las necesidades de los usuarios, determinando las actividades que se deben seguir durante la ejecución del diseño, así como la identificación de los recursos necesarios (expertos) que apoyarán la construcción del diseño.

Se debe tener presente si se cuenta con experiencia previa en proyectos iguales o similares con el fin de reutilizar definiciones que puedan aplicar.

Es parte vital del aseguramiento de la calidad del diseño manejar niveles bajos de incertidumbre frente a necesidades del usuario, así como establecer alcances, objetivos y actividades, delimitándolos a través de tiempos para realizar una entrega del artefacto (diseño) que cumpla con la ejecución de un cronograma.

El alcance define el control del diseño, incluye lo necesario para llegar a una culminación exitosa, contiene un detalle claro de la necesidad a resolver, permitiendo tener un objetivo preciso, facilitando alcanzar las metas propuestas.

La planeación disminuye al mínimo los problemas potenciales y proporciona un mayor rendimiento de tiempos y esfuerzo.

De igual manera, el correcto planteamiento de objetivo del diseño debe hacerse de manera clara, concreta, y alcanzable, es decir realizable o que se pueda llevar a la práctica.

Las decisiones que sean tomadas durante esta fase permiten eliminar la improvisación, es decir evaluar alternativas antes de tomar una decisión.

6.1.1 Artefactos de entrada

- Requisitos: Se reciben el documento de requisitos definido y aprobado por los usuario.
- Casos de uso: Se reciben el documento de casos de uso con los diagramas que ilustren los requisitos elicitados en conjunto con el usuario.
- Diseño detallado de arquitectura: Se reciben el documento de diseño detallado con los modelos de la estructura global del sistema.

6.1.2 Artefactos de salida

- Requisitos ajustados: Si en las mesas de trabajo se realizan sugerencias de ajuste a los requisitos para un mayor entendimiento de los mismos, estos se deben ajustar según dichas sugerencias.
- Casos de uso ajustados: Al tener una relación directa con los requisitos, si estos son ajustados, de igual manera se deben realizar ajustes a los casos de uso.
- Diseño detallado de arquitectura ajustado: Al tener una relación directa con los requisitos, si estos son ajustados, de igual manera se debe ajustar el diseño detallado de arquitectura.
- Actas de reunión: Se deben realizar un acta de reunión por cada una de las mesas de trabajo que se realicen, ya sean con los stakeholders como con el equipo de TI.

  El acta debe contener el tema tratado en la mesa de trabajo, la fecha en la que se realizó la mesa de trabajo, los participantes a la misma y su porcentaje de participación, el orden de los temas tratados, como las conclusiones a las que se llegaron, y si se generó algún pendiente que deba ser resulto, se debe especificar dicho pendiente, así como el responsable de dar solución al mismo y la fecha en la que se dará solución.

El responsable del acta debe ser la misma persona responsable de la elaboración del diseño.

El formato de acta que se puede utilizar es el siguiente:

Formato Acta de Reunión			
Acta de reunión		**Proyecto:**	
		Acta Nro.:	
		Ciudad:	
		Lugar:	
		Fecha:	
		Hora inicio	
		Duración	
Objetivo de la reunión			
Participantes	**Porcentaje de asistencia**	**Rol**	**Correo electrónico**
1.			
2.			
3.			

4.			
n.			
Temas tratados			
Tares y compromisos			
Tarea/Actividad	**Responsable**	**Fecha compromiso**	
1.			
2.			
3.			
4.			
n.			
Responsable acta		**Fecha de elaboración**	

Formato 1 Formato de acta para reuniones

6.2 Hacer

Ejecución de las tareas exactamente de la forma prevista en el plan.

Esta etapa tiene como finalidad entregar el documento de diseño según los lineamientos recolectados en la etapa anterior (planear).

En esta etapa se debe considera realizar mesas de trabajo con el equipo de TI, donde se haga una puesta en común de las necesidades del usuario y de los objetivos de la solución.

Así mismo, en esta mesa de trabajo se debe poner en común los las necesidades del usuario de tal manera que se trabaje en conjunto para lograr alcanzarlas.

El aseguramiento de la calidad en el diseño se llega a través de los instrumentos que contienen los indicadores de calidad que se obtuvieron teniendo en cuenta las características de la familia de normas de calidad ISO/IEC 25000.

Para la elaboración del documento de diseño se procede a identificar las características de la familia de normas de calidad ISO/IEC 25000 que ayuden a robustecerlo.

Aseguramiento de la calidad en el diseño del software	**Funcionalidad**	**Adecuación:** La capacidad del diseño para proveer un adecuado conjunto de funciones para las tareas y objetivos especificados por el usuario. **Exactitud:** La capacidad del diseño para proveer los resultados o efectos acordados con un grado necesario de precisión. **Conformidad con la funcionalidad:** La capacidad del diseño de adherirse a los estándares, convenciones o regulaciones legales y prescripciones similares referentes a la funcionalidad.

	Interoperabilidad	**Compatibilidad:** La capacidad del diseño para ser cooperativamente operable con uno o más diseños. **Conformidad con la interoperabilidad:** La capacidad del diseño para adherirse a los estándares, convenciones o regulaciones en lo relacionado a la interoperabilidad.
	Fiabilidad	**Madurez:** La capacidad del diseño para la fácil detección de inconsistencia y corrección de las mismas. **Tolerancia a fallos:** La capacidad del diseño para controlar las inconsistencia encontradas y mantener un nivel especificado. **Conformidad con la fiabilidad:** La capacidad del diseño para adherirse a las normas, convenciones o regulaciones relativas a la fiabilidad.
	Usabilidad	**Apropiabilidad:** La capacidad del diseño para permitir al usuario entender si el diseño es adecuado, y cómo puede ser utilizado y las condiciones particulares del diseño. **Conformidad del uso:** La capacidad del diseño para adherirse a los estándares, convenciones, guías de estilo o regulaciones relacionadas a la usabilidad.

	Eficiencia	**Comportamiento en el tiempo:** La capacidad del diseño para proveer tiempos adecuados de respuesta en concordancia con la necesidad del usuario. **Conformidad con la eficiencia:** La capacidad del diseño para adherirse a estándares o convenciones relacionados con la eficiencia.
	Mantenibilidad	**Capacidad de ser analizado:** La capacidad del diseño para ser diagnosticado por deficiencias o causas de fallas en el diseño o la identificación de las partes a ser modificadas. **Facilidad de cambio:** La capacidad del diseño para permitir que una determinada modificación sea implementada. **Estabilidad:** La capacidad del diseño para evitar efectos inesperados debido a modificaciones del diseño. **Conformidad con la facilidad de mantenimiento:** La capacidad del diseño para adherirse a estándares o convenciones relativas a la facilidad de mantenimiento.

	Portabilidad	**Adaptabilidad:** La capacidad del diseño para ser adaptado a diferentes entornos especificados sin aplicar acciones o medios diferentes de los previstos para el propósito del diseño considerado. **Reemplazabilidad:** La capacidad del diseño para ser utilizado en lugar de otro diseño, para el mismo propósito y en el mismo entorno. **Conformidad con la portabilidad:** La capacidad del diseño para adherirse a estándares o convenciones relacionados a la portabilidad.

Tabla 1: Aseguramiento de la calidad en el diseño

Las características de la tabla anterior se obtuvieron a través de los indicadores de calidad de la familia de normas ISO/IEC 25000, y deben ser aplicados por la persona responsable del diseño, estas tiene como fin construir un diseño en términos de Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.

El tiempo a emplear para la construcción del diseño debe ser determinado por el proyecto y lo que se indique puntualmente para esta etapa.

6.2.1 Artefactos de entrada

- Requisitos.
- Casos de uso.
- Diseño detallado de arquitectura.

6.2.2 Artefactos de salida

- Documento de diseño: Contiene en detalle la descripción de las características del sistema que se debe construir y que resuelve las necesidades planteadas por el usuario en el documento de requisitos.

6.3 Verificar:

Compara el resultado obtenido con la meta planificada.

Una vez finalizado el diseño obtenido con los diferentes instrumentos aplicados en la etapa anterior (hacer), se debe realizar una mesa de trabajo con el equipo de desarrollo para revisar el diseño, con el objetivo de identificar las restricciones que la herramienta o lenguaje de programación puedan presentar frente al diseño construido.

Según lo anterior se debe tener presente que el diseño puede se ajustado según la restricciones encontradas y nuevas definiciones que se identifiquen con el usuarios, en este orden de ideas el diseño debe ser los suficientemente flexible para soportar los cambios.

Posteriormente se debe realizar una mesa de trabajo con los stakeholders, que tendrá como objetivo revisar el diseño y verificar en conjunto con los usuarios que se cumplan las expectativas de las necesidades plantadas en los requisitos.

Como resultado de esta mesa de trabajo se debe generar un acta con los resultados de la misma.

6.3.1 Artefactos de entrada

- Documento de diseño.

6.3.2 Artefactos de salida

- Actas mesas de trabajo: Actas con el detalle de las reuniones realizadas con los desarrolladores, así como con los stakeholders. Estas actas consideran, si es necesario realizar algún ajuste al diseño.

6.4 Actuar:

Ajustes para alcanzar objetivos.

Corresponde a las actividades que conllevan a la entrega del documento de diseño a los desarrolladores para la construcción de la solución del sistema.

En este punto ya el diseño debe contar con la calidad que asegure que cumple con las necesidades del usuario.

Si se identifica que el diseño no cumple con las necesidades del usuario, se inicia nuevamente el ciclo en la etapa de Planear, repitiendo así todas las etapas hasta alcanzar un diseño de acuerdo con los estándares aquí descritos y principalmente que cumpla con la necesidad del usuario frente al sistema.

6.4.1 Artefactos de entrada

- Actas mesas de trabajo.
- Documento de diseño.

6.4.2 Artefactos de salida

- Documento de diseño ajustado: Si en las mesas de trabajo realizadas con los stakeholders y el equipo de desarrollo se identifican ajustes al documento de diseño, es necesarios,

realizar estos ajustes para entregar un documento de diseño que cumpla con las necesidades planteadas en el documento de requisitos.

6.5 Métricas del modelo

A través del juicio de expertos se logra identificar las características, subcaracterísticas y métricas que soportan de manera adecuada el modelo propuesto en este trabajo de investigación. De este juicio de expertos sale como resultado la siguiente tabla que contiene las características, subcaracterísticas y métricas definidas para el modelo.

Característica	**Subcaracterísticas**	**Métrica**
Funcionalidad	Adecuación	Cobertura de la implementación funcional.
		Estabilidad (o volatilidad) de la especificación funcional.
		Suficiencia funcional.
		Integridad de la implementación funcional.
	Exactitud	Exactitud computacional.
		Precisión.
Interoperabilidad	Compatibilidad de datos	Intercambiabilidad de datos.
	Conformidad con la interoperabilidad	Regulación de interoperabilidad.

Fiabilidad	Madurez	Detección de fallas.
		Remoción de fallos
	Tolerancia a fallos	Prevención de fallas.
	Conformidad con la fiabilidad	Conformidad con la fiabilidad.
Usabilidad	Apropiabilidad	Integridad de la descripción.
		Funciones evidentes.
		Comprensibilidad de la función.
		Integridad de la descripción.
Eficiencia	Comportamiento en el tiempo	Tiempo de respuesta.
		Tiempo del rendimiento de procesamiento.
		Plazo de entrega.
	Conformidad con la eficiencia	Conformidad con la eficiencia.
Mantenibilidad	Capacidad de ser analizado	Preparación de la función de diagnóstico.
	Facilidad de cambio	Facilidad de registrar los cambios.
		Capacidad de control de cambio en el software .
	Estabilidad	Localización del impacto de la modificación.
		Impacto del cambio

	Conformidad con la facilidad de mantenimiento	Conformidad con la facilidad de mantenimiento.
Portabilidad	Adaptabilidad	Adaptabilidad de las estructuras de datos.
		Adaptabilidad del ambiente de hardware.
		Adaptabilidad del ambiente organizacional.
		Amigabilidad del usuario.
	Reemplazabilidad	Uso continuo de los datos.
		Inclusividad de la función.
	Conformidad Con La portabilidad	Conformidad con la portabilidad.

Tabla 2 Métricas.

6.6 Detalles de la Métricas.

A continuación se detallaran las métricas a utilizar como soporte a la medición en el aseguramiento de la calidad en el diseño del software.

6.6.1 Funcionalidad.

6.6.1.1 Adecuación.

Cobertura de la implementación funcional.

Nombre:	**Functional implementation coverage**

Propósito:	How correct is the functional implementation?
Método de aplicación:	Count the number of incorrectly implemented or missing functions and compare with the number of functions described in the requirement specifications.
Medición, fórmula:	X = 1-A/B A = Number of incorrectly implemented or missing functions detected B = Number of functions described in requirement specifications
Interpretación:	0 <= X <= 1 The closer to 1, the more correct
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 3 Cobertura de la implementación funcional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Estabilidad (o volatilidad) de la especificación funcional.

Nombre:	**Functional specification stability (volatility)**
Propósito:	How stable is the functional specification during the development life cycle?
Método de	Count the number of functions changed (added, modified, or deleted)

aplicación:	during development life cycle phase, then compare with the number of functions described in the requirement specifications.
Medición, fórmula:	X = 1-A/B A = Number of functions changed during development life cycle phases B = Number of functions described in requirement specifications
Interpretación:	0 <= X <= 1 The closer to 1, the more stable
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 4 Estabilidad (o volatilidad) de la especificación funcional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Suficiencia funcional.

Nombre:	**Functional adequacy**
Propósito:	How adequate are the evaluated functions?
Método de aplicación:	Number of functions that are suitable for performing the specified tasks comparing to the number of function evaluated.

Medición, fórmula:	X = 1-A/B A = Number of functions in which problems are detected in evaluation B = Number of functions evaluated
Interpretación:	0 <= X <= 1 The closer to 1.0, the more adequate
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 5 Suficiencia funcional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Integridad de la implementación funcional.

Nombre:	**Functional implementation completeness**
Propósito:	How complete is the implementation according to requirement specifications?
Método de aplicación:	Do functional tests (black box test) of the system according to the requirement specifications. Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications.

Medición, fórmula:	X = 1-A/B A = Number of missing functions detected in evaluation B = Number of functions described in requirement specifications
Interpretación:	0 <= X <= 1 The closer to 1.0, is the better.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 6 Integridad de la implementación funcional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.1.2 Exactitud.

Exactitud computacional.

Nombre:	**Computational accuracy**
Propósito:	How often do the end users encounter inaccurate results?
Método de aplicación:	Record the number of inaccurate computations based on specifications.
Medición, fórmula:	X = A/T A = Number of inaccurate computations encountered by users T = Operation time

Interpretación:	0 <= X The closer to 0 is the better.
Tipo de escala:	Ratio.
Tipo de medida:	X = count/time A = count T = time

Tabla 7 Exactitud computacional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Precisión.

Nombre:	**Precision**
Propósito:	How often do the end users encounter results with inadequate precision?
Método de aplicación:	Record the number of results with inadequate precision.
Medición, fórmula:	X = A/T A = Number results encountered by the users with level precision different from required. T = Operation time
Interpretación:	0 <= X The closer to 0, is the better.

Tipo de escala:	Absolute
Tipo de medida:	X = count/time A = count T = time

Tabla 8 Precisión (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.2 Interoperabilidad.

6.6.2.1 Compatibilidad de los datos.

Intercambiabilidad de Datos (Basado en el formato de datos).

Nombre:	**Data exchangeability (Data format based)**
Propósito:	How correctly have the interface data formats been implemented?
Método de aplicación:	Count the number of interface data formats that have been implemented correctly as in the specifications and compare to the number of data formats to be exchanged as in the specifications.
Medición, fórmula:	X = A/B A = Number of interface data formats that have been implemented correctly as in the specifications. B = Number of data formats to be exchanged as in the specifications.
Interpretación:	0 <= X <= 1

	The closer to 1, the more correct.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Requirement specification Design Source code Review Report
ISO/IEC 12207 **SLCP:**	Verification Joint review
Audiencia:	Developers Requirers

Tabla 9 Intercambiabilidad de datos (Basado en el formato de datos) (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.2.2 Conformidad con la interoperabilidad.

Regulación de interoperabilidad.

Nombre:	**Interoperability regulation**
Propósito:	How many interopperability regulations are being met.

Método de aplicación:	Evaluate the number of interoperability regulations that are being met and compare to the number of interoperability regulations that should be met according to the specifications.
Medición, fórmula:	X = A/B A = Number of interoperability regulations that are being met. B = Number of interoperability regulations that should be met according to the specifications.
Interpretación:	0 <= X <= 1 6.6.2.2.1.1 The closer to 1, is the better.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = coun B = coun

Tabla 10 Regulación de interoperabilidad (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.3 Fiabilidad.

6.6.3.1 Madurez.

Detección de fallas.

Nombre:	**Fault detection**

Propósito:	How many faults were detected in reviewed product?
Método de aplicación:	Count the number of detected faults in review and compare it to the number of estimated faults to be detected in this phase.
Medición, fórmula:	X = A/B A = Absolute number of faults detected in review B = Number of estimated faults to be detected in review (using past history or reference model)
Interpretación:	0 <= X A high value for X implies good product quality, while A=0 does not necessarily imply fault free status of the reviewed item.
Tipo de escala:	Absolute.
Tipo de medida:	X = count/count A = count B = count

Tabla 11 Detección de fallas (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Remoción de fallos.

Nombre:	**Fault removal**
Propósito:	How many faults have been corrected?

	What is the proportion of faults removed?
Método de aplicación:	Count the number of faults removed during design/coding and compare it to the number of faults detected in review during design/coding.
Medición, fórmula:	X = A A = Number of corrected faults in design/coding Y = A/B A = Number of corrected faults in design/coding. B = Number of faults detected in review.
Interpretación:	0 <= X A high value for X implies, that less faults remain 0 <= Y <= 1 The closer to 1, the better (more faults removed)
Tipo de escala:	Ratio. Absolute.
Tipo de medida:	Y = count/count B = count

Tabla 12 Remoción de fallos (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.3.2 Tolerancia a fallos.

Prevención de fallas.

Nombre:	**Failure avoidance**
Propósito:	How many fault patterns were brought under control to avoid critical and serious failures?
Método de aplicación:	Count the number of avoided fault patterns and compare it to the number of fault patterns to be considered.
Medición, fórmula:	X = A/B A = Number of fault patterns having avoidance in design/code B = Number of fault patterns to be considered ***Comment(s)*** Fault pattern examples out of range data deadlock ***Comment(s)*** Fault tree analysis technique may be used to detected fault patterns.
Interpretación:	0 <= X Where X is greater the better failure avoidance.
Tipo de escala:	Absolute.
Tipo de medida:	X = count/count A = count B = count

Tabla 13 Prevención de fallas (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.3.3 Conformidad con la fiabilidad.

Métrica externa de la conformidad con la fiabilidad - conformidad con la fiabilidad.

Nombre:	**Reliability compliance**
Propósito:	How compliant is the reliability of the product to applicable regulations, standards and conventions?
Método de aplicación:	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification
Medición, fórmula:	X = 1- A/B A = Number of reliability compliance items specified that have not been implemented during testing B = Total number of reliability compliance items specified
Interpretación:	0 <= X <= 1 The closer to 1.0, is the better.
Tipo de escala:	Absolute.
Tipo de medida:	X = count/count A = count B = count

Tabla 14 Conformidad con la fiabilidad (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.4 Usabilidad.

6.6.4.1 Apropiabilidad.

Integridad de la descripción.

Nombre:	**Completeness of description**
Propósito:	What proportion of functions (or types of function) are described in the product description?
Método de aplicación:	Count the number of functions which are adequately described and compare with the total number of functions in the product.
Medición, fórmula:	X = A/B A = Number of functions (or types of functions) described in the product description B = Total number of functions (or types of functions)
Interpretación:	0 <= X <= 1 The closer to 1, the more complete.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 15 Integridad de la descripción (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Funciones evidentes.

Nombre:	**Evident functions**
Propósito:	What proportion of the product functions are evident to the user?
Método de aplicación:	Count the number of functions that are evident to the user and compare with the total number of functions.
Medición, fórmula:	X = A/B A = Number of functions (or types of functions) evident to the user B = Total number of functions (or types of functions)
Interpretación:	0 <= X <= 1 The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 16 Funciones evidentes (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Comprensibilidad de la función.

Nombre:	**Function understandability**
Propósito:	What proportion of the product functions will the user be able to

	understand correctly?
Método de aplicación:	Count the number of user interface functions where purposes is understood by the user and compare with the number of user interface functions.
Medición, fórmula:	X = A/B A = Number of user interface functions whose purpose is understood by the user B = Number of user interface functions
Interpretación:	0 <= X <= 1 The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 17 Comprensibilidad de la función (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Integridad de la descripción.

Nombre:	**Completeness of description**
Propósito:	What proportion of functions (or types of function) is understood after reading the product description?

Método de aplicación:	Conduct user test and interview user with questionnaires or observe user behavior. Count the number of functions which are adequately understood and compare with the total number of functions in the product.
Medición, fórmula:	X = A/B A = Number of functions (or types of functions) understood. B = Total number of functions (or types of functions)
Interpretación:	0 <= X <= 1 The closer to 1.0 is the better.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count

Tabla 18 Integridad de la descripción (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.4.2 Conformidad de uso.

Conformidad con la usabilidad.

Nombre:	**Usability compliance**
Propósito:	How compliant is the product to applicable regulations, standards and conventions for usability?

Método de aplicación:	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.
Medición, fórmula:	X = A/B A = Number of correctly implemented items related to usability compliance confirmed in Evaluation B = Total number of compliance ítems
Interpretación:	0 <= X <= 1 The closer to 1, the more compliant.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Specification of compliance and related standards, conventions or regulations. Design Source code Review report

Tabla 19 Conformidad con la usabilidad (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.5 Eficiencia.

6.6.5.1 Comportamiento en el tiempo.

Tiempo de respuesta.

Nombre:	**Response time**
Propósito:	What is the estimated time to complete a specified task?
Método de aplicación:	Evaluate the efficiency of the operating system and the application system calls. Estimate the response time based on this. The following may be measured, - all or parts of design specifications - test complete transaction path - test complete modules/parts of software product - complete software product during test phase
Medición, fórmula:	X = time (calculated or simulated)
Interpretación:	The shorter the better
Tipo de escala:	Ratio
Tipo de medida:	X = time
Fuente de medición:	Known operating system. Estimated time in system calls.

Tabla 20 Tiempo de respuesta (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Tiempo del rendimiento de procesamiento.

Nombre:	**Throughput time**
Propósito:	What is the estimated number of tasks that can be performed over a unit of time?
Método de aplicación:	Evaluate the efficiency of handling resources in the system. Make a factor based upon the application calls to the system in handling the resources.
Medición, fórmula:	X = Number of tasks per unit of time
Interpretación:	The greater the better
Tipo de escala:	Ratio
Tipo de medida:	X = count
Fuente de medición:	Known operating system. Estimated time in system calls.

Tabla 21 Tiempo del rendimiento de procesamiento (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Plazo de entrega.

Nombre:	**Turnaround time**
Propósito:	What is the estimated time to complete a group of related tasks as job lot?
Método de	Evaluate the efficiency of the operating system and the application

aplicación:	system calls. Estimate the response time to complete a group of related tasks based on this. The following may be measured, - all or parts of design specifications - test complete transaction path - test complete modules/parts of software product - complete software product during test phase
Medición, fórmula:	X = time (calculated or simulated)
Interpretación:	The shorter the better
Tipo de escala:	Ratio
Tipo de medida:	X = time
Fuente de medición:	Known operating system. Estimated time in system calls.

Tabla 22 Plazo de entrega (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.5.2 Conformidad con la eficiencia.

Conformidad con la eficiencia.

Nombre:	**Efficiency compliance**
Propósito:	How compliant is the efficiency of the product to applicable

	regulations, standards and conventions?
Método de aplicación:	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.
Medición, fórmula:	X = A/B A = Number of correctly implemented items related to efficiency compliance confirmed in evaluation B = Total number of compliance items
Interpretación:	0 <= X <= 1 The closer to 1, the more compliant.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Specification of compliance and related standards, conventions or regulations Design Source code Review report

Tabla 23 Conformidad con la eficiencia (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.6 Mantenibilidad.

6.6.6.1 Capacidad de ser analizado.

Preparación de la función de diagnóstico.

Nombre:	**Readiness of diagnostic function**
Propósito:	How thorough is the provision of the diagnostic functions?
Método de aplicación:	Count the number of implemented diagnostic functions as specified and compare it to the number of diagnostic functions required in specifications.
Medición, fórmula:	X = A/B A = Number of implemented diagnostic functions as specified confirmed in review B = Number of diagnostic functions required
Interpretación:	0 <= X The closer to 1, the better implementation of diagnostic ***Comment(s)*** It is necessary to convert this value to the <0,1> interval if making summarization of characteristics.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Value A comes from review report Value B comes from requirement specifications

Tabla 24 Preparación de la función de diagnóstico (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.6.2 Facilidad de cambio.

Facilidad de registrar los cambios.

Nombre:	**Change recordability**
Propósito:	Are changes to specifications and program modules recorded adequately in the code with comment lines?
Método de aplicación:	Record ratio of module change information.
Medición, fórmula:	X = bits/time (calculated or simulated)
Interpretación:	0 <= X <= 1 The closer to 1, the more recordable. The charge control 0 indicates poor change control or little changes, high stability.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Configuration control system Version logs

	Specifications

Tabla 25 Facilidad de registrar los cambios (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Capacidad de control de cambio en el software.

Nombre:	**Software change control capability**
Propósito:	Can the user easily identify revised versions? Can the maintainer easily change the software to resolve problems?
Método de aplicación:	Observe the behaviour of user or maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report
Medición, fórmula:	X = A/B A = Number of change log data actually recorded B = Number of change log data planned to be recorded enough to trace software changes.
Interpretación:	0 <= X <= 1 The closer to 1, is the better or the closer to 0 the fewer changes have taken place.
Tipo de escala:	Absolute
Tipo de medida:	A = count

	B = count X = count/count
Fuente de medición:	User manual or specification Problem resolution report Maintenance report Operation report

Tabla 26 Capacidad de control de cambio en el software (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.6.3 Estabilidad.

Localización del impacto de la modificación.

Nombre:	**Modification impact localization**
Propósito:	How large is the impact of the modification on the software product?
Método de aplicación:	Count the number of affected variables from a modification and compare it to the total number of variables in the product. ***Comment(s)*** Impacted variable is a) All variables in the instruction which was changed b) Variable which is in the same instruction with the variable defined by a)
Medición, fórmula:	X = A/B A = Number of affected variable data by modification, confirmed in

	review. B = Total number of variables.
Interpretación:	0 <= X <= 1 The closer to 1, the lesser impact of modification
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	A comes from review report B comes from review report

Tabla 27 Localización del impacto de la modificación (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Impacto del cambio.

Nombre:	**6.6.6.3.1.1 Impact of the change**
Propósito:	Can user operate software system without failures after maintenance? Can maintainer easily mitigate failures caused by maintenance side effects?
Método de aplicación:	Count failures occurrences after change, which are mutually chaining and affected by change.
Medición, fórmula:	X = A/N

	A = Number of failures emerged after failure is resolved by change during specified period N = Number of resolved failures
Interpretación:	0 <= X The smaller and closer to 0, is the better.
Tipo de escala:	Absolute
Tipo de medida:	A = count N = count X = count/count
Fuente de medición:	Problem resolution report Operation report

Tabla 28 Impacto del cambio (Fallo que emerge después del cambio) (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.6.4 Conformidad con la facilidad de mantenimiento.

Conformidad con la facilidad de mantenimiento.

Nombre:	**Maintainability compliance**
Propósito:	How compliant is the maintainability of the product to applicable regulations, standards and conventions?

Método de aplicación:	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.
Medición, fórmula:	X = A/B A = Number of correctly implemented items related to maintainability compliance confirmed in evaluation B = Total number of compliance items
Interpretación:	0 <= X <= 1 The closer to 1, the more compliant.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Specification of compliance and related standards, conventions or regulations Design Source code Review report

Tabla 29 Conformidad con la facilidad de mantenimiento (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.7 Portabilidad.

6.6.7.1 Adaptabilidad.

Adaptabilidad de las estructuras de datos.

Nombre:	**Adaptability of data structures**
Propósito:	How adaptable is the product to the data structure changes?
Método de aplicación:	Count the number of data structures, which are operable and has no limitation after adaptation and compare it to the total number of data structures requiring adaptation capability
Medición, fórmula:	X = A/B A = Number of data structures which are operable and has no limitation after adaptation, confirmed in review B = Total number of data structures requiring adaptation capability
Interpretación:	0 <= X <= 1 The closer to 1, the better.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Requirements specifications Design Review report

Tabla 30 Adaptabilidad de las estructuras de datos (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Adaptabilidad del ambiente de hardware.

Nombre:	**Hardware environmental adaptability**
Propósito:	How adaptable is the product to the H/W related environmental change?
Método de aplicación:	Count the number of implemented functions which are capable of achieving required results in specified multiple H/W environments as specified and compare it to the number of functions with H/W environment adaptation capability requirements
Medición, fórmula:	X = A/B A = Number of implemented functions which are capable of achieving required results in specified multiple H/W environment as specified, confirmed in review B = Total number of functions with H/W environment adaptation capability requirements
Interpretación:	0 <= X <= 1 The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count

	B = count
Fuente de medición:	Requirements specifications Design Review report

Tabla 31 Adaptabilidad del ambiente de hardware (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Adaptabilidad del ambiente organizacional.

Nombre:	**Organizational environment adaptability**
Propósito:	How adaptable is the product to organizational change?
Método de aplicación:	Count the number of implemented functions which are capable of achieving required results in specified multiple organizational and business environments as specified and compare it to the number of functions with organizational environment adaptation capability requirements.
Medición, fórmula:	X = A/B A = Number of implemented functions which are capable of achieving required results in specified multiple organizational and business environment as specified, confirmed in review B = Total number of functions with organizational environment adaptation capability requirements

Interpretación:	0 <= X <= 1 The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Requirements specifications Design Review report

Tabla 32 Adaptabilidad del ambiente organizacional (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Amigabilidad del usuario.

Nombre:	**Porting user friendliness**
Propósito:	Can user or maintainer easily adapt software to environment?
Método de aplicación:	Observe user's or maintainer's behaviour when user is trying to adapt software to operational environment.
Medición, fórmula:	T = Sum of user operating time spent to complete adaptation of the software to user's environment, when user attempt to install or change setup.
Interpretación:	0 < T

	The shorter is the better.
Tipo de escala:	Ratio
Tipo de medida:	T= time
Fuente de medición:	Problem resolution report Operation report

Tabla 33 Amigabilidad del usuario (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.7.2 Reemplazabilidad.

Uso continuo de los datos.

Nombre:	**Continued use of data**
Propósito:	What is the amount of original data that remain unchanged after replacement with this product?
Método de aplicación:	Count the number of data items, that continue to be used after replacement as specified, and compare it to the total number of data items required to be used from the old data after software replacement.
Medición, fórmula:	X = A/B A = Number of software data items that continue to be used as specified after replacement, confirmed in evaluation B = Number of old data items required to be used from old software.
Interpretación:	0 <= X <= 1

	The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Design Source code Review report Test report

Tabla 34 Uso continuo de los datos (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

Inclusividad de la función.

Nombre:	**Function inclusiveness**
Propósito:	What's the amount of functions that remain unchanged?
Método de aplicación:	Count the number of functions covered by new software that produces similar results and compare it to the number of function in the old software.
Medición, fórmula:	X = A/B A = Number of functions covered by new software that produces similar results confirmed in review.

	B = Number of functions in old software.
Interpretación:	0 <= X <= 1 The closer to 1, the better
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Design Source code Review report Test report

Tabla 35 Inclusividad de la función (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

6.6.7.3 Conformidad con la portabilidad.

Métrica externa de la conformidad con la portabilidad - conformidad con la portabilidad.

Nombre:	**Portability compliance**
Propósito:	How compliant is the portability of the product to applicable regulations, standards and conventions?
Método de aplicación:	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the

	specification.
Medición, fórmula:	X = 1- A/B A = Number of portability compliance items specified that have not been implemented during testing B = Total number of portability compliance items specified.
Interpretación:	0 <= X <= 1 The closer to 1, is the better.
Tipo de escala:	Absolute
Tipo de medida:	X = count/count A = count B = count
Fuente de medición:	Product description (User manual or Specification) of compliance and related standards, conventions or regulations. Testing specification and report

Tabla 36 Conformidad con la portabilidad (Fuente: Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007).

7 Ejemplificación del modelo.

A continuación se realiza un caso de ejemplo de la utilización del modelo aplicando las característica, subcaracterística y métricas definidas para el aseguramiento de la calidad en el diseño del software.

Para la ejemplificación del modelo se plantea la aplicación de la norma ISO/IEC 25000.
Se deben establecer las características, subcaracterísticas y métricas a evaluar con sus valores respectivos requeridos por el cliente.
En este paso, se deberá:

- Establecer un valor especificado por el cliente para cada métrica.
- Establecer un valor especificado por el cliente para cada subcaracterística.
- Establecer un valor especificado por el cliente para cada característica.
- En caso que la característica, subcaracterística o métrica no desee ser evaluada, el valor especificado por el cliente será cero.

En el caso de la evaluación de las métricas, se considerará que una métrica se cumple cuando el resultado de la medición es mayor o igual al valor especificado por el cliente.
El resultado de la medición surge del producto entre y la aplicación de la fórmula de la métrica que corresponda.
En la evaluación de la subcaracterística, se considerará que la subcaracterística se cumple cuando el promedio ponderado de sus métricas respectivas es mayor o igual al valor especificado por el cliente. Este promedio ponderado se basa en los resultados de las mediciones pertenecientes a las métricas de una subcaracterística determinada.
En la evaluación de la característica, se considerará que la característica se cumple cuando el

promedio ponderado de sus subcaracterísticas respectivas es mayor o igual al valor especificado por el cliente. Este promedio ponderado se basa en los promedios ponderados de las subcaracterísticas pertenecientes a una característica determinada.

Por cada métrica que se desee evaluar, se calcula el valor de la medición y el resultado de la medición.

El valor de la medición surge de la aplicación de la fórmula de la métrica establecida en la norma ISO/IEC 25000. Este valor es solo a nivel métrica.

El resultado de la medición surge del producto del valor de la medición por el valor especificado por el cliente.

Por cada Subcaracterística, se debe calcular el promedio ponderado basado en los resultados de las mediciones de las métricas respectivas. Por cada Característica, se debe calcular el promedio ponderado basado en las promedios de las subcaracterísticas respectivas.

Valores Respectivos Requeridos por el Cliente →VRRC.

Resultado de la Medición → RM.

Fórmula de la Métrica → FM.

Promedio Ponderado de sus Métricas → PPM.

7.1 Funcionalidad.

7.1.1 Adecuación.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Adecuación" es de PPM = 0.1825. De acuerdo al valor especificado por el cliente VRRC = 0.30 , esta subcaracterística no se cumple de manera satisfactoria.

7.1.1.1 Alcance de la implementación funcional.

Qué tan correcta es la implementación del diseño?

X = 1 - A/B

A = número de funciones faltantes en el diseño.

B = número de funciones descritas en la especificación de requisitos.

En el análisis del diseño, se determina que hay 3 funciones faltantes (A=3) y que la cantidad de funciones existentes coincide con lo especificado en los requerimientos (B=10). Por lo tanto, el "Alcance de la implementación funcional" es FR = 0.70 (1-A/B). De acuerdo al valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.14.

7.1.1.2 Estabilidad (o volatilidad) de la especificación funcional.

Qué tan estable es el diseño durante el desarrollo del ciclo de vida?

X = 1 - A/B

A = número de funciones cambiadas durante las fases de desarrollo del ciclo de vida.

B = número de funciones descritas en la especificación de requisitos.

En el diseño 4 funciones sufrieron cambios y/o actualizaciones durante las etapas del ciclo de vida de desarrollo (A= 4). La especificación de requerimientos describe 10 funciones (B=10). Por lo tanto, el "Estabilidad de la especificación funcional" es FM = 0.60 (1-A/B).

De acuerdo al valor especificado por el cliente VRRC = 0.30, el resultado final es RM = 0.18.

7.1.1.3 Suficiencia funcional.

Que tan adecuadas son las funciones verificadas?

X = 1 - A/B

A = número de funciones detectadas con problemas en el diseño.

B = número de funciones verificadas.

En el diseño se detectó que 3 funciones tienen problemas (A=3) de un total de 10 funciones controladas (B=10). Por lo tanto, la "Suficiencia funcional" es FM = 0.7 (1-A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.30, el resultado final es RM = 0.21.

7.1.1.4 Integridad de la implementación funcional

Qué tan completo está el diseño de acuerdo a la especificación de requisitos?

X = 1-A/B

A = número de funciones faltantes.

B = número de funciones descritas en la especificación de requisitos.

En el diseño se detectó que no hay funciones ausentes (A=0) y que

la cantidad de funciones existentes coincide con las especificadas en los requerimientos (B=10). Por lo tanto, la "Integridad de la implementación funcional" es FM = 1 (1-A/B). De acuerdo al valor especificado por el cliente VRRC = 0.20, el resultado final es RM= 0.20.

7.1.2 Exactitud.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Exactitud" es PPM = 0.38. De acuerdo al valor especificado por el cliente VRRC = 0.30, esta subcaracterística se cumple de manera satisfactoria.

7.1.2.1 Exactitud en el diseño.

Cuantas veces se encuentran resultados inexactos en el diseño?

X = A/B

A = Número de cómputos inexactos encontrados por los usuarios.

B = Funciones que requieren de esta particularidad.

En el diseño, se consideraron los requerimientos de exactitud específica en 7 funciones (A=7) de un total de 10 funciones que requieren de esta particularidad (B=10). Por lo tanto, la "Exactitud en el diseño " es FM = 0.7 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.80, el resultado final es RM = 0.56.

7.1.2.2 Precisión.

Cuantas veces se encuentran resultados con precisión inadecuada en el diseño?

X = A/B

A = Número de funciones encontrados en el diseño con nivel de precisión distinta de la requerida.

B = Número de funciones que requieren de niveles de precisión específicos.

Existen 10 funciones en el diseño con ciertos niveles de precisión especificados (A=10). Dicha cantidad coincide con el número de funciones que requieren de niveles de precisión específicos (B=10). Por lo tanto, la "Precisión" es FM = 1 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.20.

7.1.3 Conformidad con la funcionalidad.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Conformidad" es PPM = 0.16. De acuerdo al valor especificado por el cliente VRRC = 0.30, esta subcaracterística no cumple de manera satisfactoria.

7.2 Interoperabilidad.

7.2.1 Compatibilidad de los datos.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Compatibilidad de los datos " es PPM = 0.60. De acuerdo al valor especificado por el cliente VRRC = 0.10, esta subcaracterística se cumple de manera satisfactoria.

7.2.1.1 Intercambiabilidad de datos (basado en el formato de datos).

Que tan correctamente han sido diseñado los formatos de la interfaz de datos?

X = A/B

A = Número de formatos de la interfaz de datos que han sido diseñados correctamente.

B = Número de formato de datos a ser intercambiados según los requisitos.

En el diseño de se especificó un formato de datos de interface (A=1). Este formato de datos puede ser cambiado, en caso que la situación lo requiera (B=1). Por lo tanto, la "Intercambiabilidad de datos" es FM = 1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.60, el resultado final es RM = 0.60.

7.2.2 Conformidad con la interoperabilidad.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Compatibilidad de los datos " es PPM = 0.40. De acuerdo al valor especificado por el cliente VRRC = 0.10, esta subcaracterística se cumple de manera satisfactoria.

7.2.2.1 Regulación de interoperabilidad.

Cuantas regulaciones de interoperabilidad se están cumpliendo?

X = A/B

A = Número de regulaciones de interoperabilidad que se cumplen en el diseño.

B = Número de regulaciones de interoperabilidad que se deben cumplir según los requisitos.

El diseño cuenta con la una regulación (A=1) y esta especificada en los requerimientos (B=1). Por lo tanto, la "Regulación de interoperabilidad " es FM = 1 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.40.

7.3 Fiabilidad.

7.3.1 Madurez.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Madurez" es de PPM = 0.06. De acuerdo al valor especificado por el cliente VRRC = 0.40, esta subcaracterística no se cumple de manera satisfactoria.

7.3.1.1 Detección de fallas.

Cuantas fallas fueron detectadas en el diseño?

X = A/B

A = Número absoluto de fallos detectados en diseño.

B = Número de fallos estimados a ser detectados en el diseño.

Durante la revisión del diseño se detectaron 3 defectos (A=3) de un total de 10 defectos estimados (B=10). Por lo tanto, la "Detección de defectos" es FM = 0.3 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.40 , el resultado final es RM = 0.12.

7.3.1.2 Remoción de fallos.

Cuál es la proporción de fallos removidos?

X = A/B

A = Número de fallos corregidos en el diseño.

B = Número de fallos detectados en la revisión.

En el diseño no fueron corregidos (A=0), y fueron detectados 3 en la revisión (B=3). Por lo tanto, la "Remoción de fallos" es FM = 0 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.

7.3.2 Tolerancia a fallos.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Tolerancia a fallas" es de PPM = 1.08. De acuerdo al valor especificado por el cliente VRRC = 0.30, esta subcaracterística se cumple de manera satisfactoria.

7.3.2.1 Prevención de fallas.

Cuantos patrones de fallos fueron traídos bajo control para evitar fallas criticas y serias?

X = A/B

A = Número de patrones de fallas teniendo evasión en el diseño.

B = Número de patrones de fallas a ser consideradas.

Se estableció que 3 funciones tienen problemas, las cuales tienen un modelo de falla asociado cada una (B=3). En la etapa de diseño se determinaron que pueden suceder 3 modelos de fallas, las cuales hacen referencia al ingreso de datos, procesamiento y salida de datos (A=3). Por lo tanto, la "Prevención de fallas" es FM = 1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.50, el resultado final es RM = 0.50.

7.3.2.2 Prevención de operación incorrecta.

Cuantas funciones preventivas paras las operaciones incorrectas están diseñadas?

X = A/B

A = Número de interrupciones.

B = Número de faltas.

En este caso se tienen 10 funciones diseñadas que evitan operaciones incorrectas (A=10) y se tiene como referencia que existen 3 modelos de operación incorrectos (B=3). Por lo tanto, la "Prevención de operación incorrecta" es FM = 3.34 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.50, el resultado final es RM = 1.67.

7.3.3 Conformidad con la fiabilidad.

Esta Subcaracterística no es evaluada, VRRC = 0.

7.3.3.1 Conformidad con la fiabilidad.

Qué tan conforme es la fiabilidad del diseño para aplicar regulaciones, estándares y convenciones?

X = 1- A/B

A = Número de datos de la conformidad de la fiabilidad diseñados.

B = Número total de datos de la conformidad de la fiabilidad especificados en el requerimiento.

7.4 Usabilidad.

7.4.1 Apropiabilidad.

Teniendo en cuenta los valores calculados en las métricas anteriores, se determina que la "Facilidad de Comprensión" es de PPM = 0.198. De acuerdo al valor especificado por el cliente VRRC = 0.40, esta subcaracterística no se cumple de manera satisfactoria.

7.4.1.1 Integridad de la descripción.

Qué proporción de las funciones (o tipos de función) están descritas en el diseño?

X = A/B

A = Número de funciones (o tipos de funciones) descritas en el diseño.

B = Número total de funciones (o tipos de funciones).

Esta aplicación cuenta con la descripción de cada función asociada a un informe en particular (A=10). El número total de funciones de esta aplicación es 10 (B=10). Por lo tanto, la "Integridad de la descripción" es FM = 1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.20.

7.4.1.2 Funciones evidentes.

Qué proporción de las funciones del diseño son evidentes al usuario?

X = A/B

A = Número de funciones (o tipos de funciones) evidentes al usuario.

B = Número total de funciones (o tipos de funciones).

Las 10 funciones de este diseño pueden ubicarse sin problemas a través de la exploración de la interface. (A=10). El total de funciones es de 10 (B=10). Por lo tanto, la "Funciones evidentes" es FM = 1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.20.

7.4.1.3 Comprensibilidad de la función.

Qué proporción de las funciones del diseño estarán disponibles al usuario para ser comprendidas correctamente?

X = A/B

A = Número de funciones de interfaz de usuario cuyo propósito es entendido por el usuario.

B = Número de funciones de interfaz de usuario.

De las 10 funciones de interface de usuario de este diseño (B=10), algunas veces, 2 de ellas no son bien comprendidas (A=2). Por lo tanto, la “Facilidad de comprensión de la función” es FM =

0.20 (A/B) . Teniendo en cuenta el valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.04.

7.4.1.4 Integridad de la descripción.

Qué proporción de las funciones (o tipos de función) es comprendida después de leer el diseño del producto?

X = A/B

A = Número de funciones (o tipos de funciones) comprendidas.

B = Número total de funciones (o tipos de funciones).

De las 10 funciones (B=10), 5 de ellas no son bien comprendidas (A=5). Por lo tanto, la "Integridad de la descripción" es FM = 0.50 (A/B) . Teniendo en cuenta el valor especificado por el cliente VRRC = 0.30, el resultado final es RM = 0.15.

7.4.2 Conformidad de uso.

Esta Subcaracterística no es evaluada, VRRC = 0.

7.4.2.1 Conformidad con la usabilidad.

Qué tan conforme es el diseño según regulaciones, estándares y convenciones aplicables para la usabilidad?

X = A/B

A = Número de datos ejecutados correctamente relacionados con la conformidad de la usabilidad en el diseño.

B = Número total de datos de la conformidad.

Esta métrica no es evaluada.

VRRC = 0

7.5 Eficiencia.

7.5.1 Comportamiento en el tiempo.

Teniendo en cuenta los valores calculados en las métricas, se determina que el " Comportamiento en el tiempo" es de PPM = 0.20. De acuerdo al valor especificado por el cliente VRRC = 0.30, esta subcaracterística no cumple de manera satisfactoria.

7.5.1.1 Tiempo de respuesta.

El diseño debe contener cual es el tiempo estimado de respuesta para completar una tarea específica.

X = A/B

A = Tiempos de respuestas de cada ítem especificados en el diseño.

B = Tiempos de respuestas solicitados en los requisitos.

El diseño especifica 1ítems con su respectivos tiempos de respuesta (A=1). En los requisitos cuenta con un total de 3 ítems que deben ser especificados con tiempos de respuesta (B=3). Por lo tanto, la "Tiempo de respuesta" es FM = 0.34 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.10, el resultado final es RM = 0.034.

7.5.1.2 Tiempo del rendimiento de procesamiento.

El diseño debe contener cual es el número estimado de tareas que pueden ser realizar sobre una unidad de tiempo.

X = A/B

A = Tiempos de procesamiento de cada ítem especificados en el diseño.

B = Tiempos de procesamiento solicitado en los requisitos.

Esta métrica no es evaluada.

VRRC = 0.

7.5.1.3 Plazo de entrega.

El diseño debe contener cual es el tiempo estimado para completar un grupo de tareas de mucho trabajo.

X = A/B

A = Tiempos de procesamiento para un grupo de tareas especificados en el diseño.

B = Tiempos de procesamiento grupo de tareas solicitado en los requisitos.

El diseño especifica 6 grupos de tareas con su respectivos tiempos de respuesta (A=6). En los requisitos cuenta con un total de 8 grupo de tareas que deben ser especificados con tiempos de respuesta (B=8). Por lo tanto, la " Plazo de entrega " es FM = 0.34 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.50, el resultado final es RM = 0.375.

7.5.2 Utilización de recursos.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Utilización de recursos" es de PPM = 0.50. De acuerdo al valor especificado por el cliente VRRC = 0.30, esta subcaracterística se cumple de manera satisfactoria.

7.5.2.1 Utilización de entrada/salida.

El diseño debe contener cual es la utilización estimada de entrada/salida para completar una tarea específica.

X = A/B

A = Utilización recursos especificados en el diseño.

B = Utilización recursos solicitado en los requisitos.

Esta métrica no es evaluada.

VRRC = 0.

7.5.2.2 Utilización de la memoria.

El diseño debe contener cual es el tamaño estimado de la memoria que el producto ocupará para terminar una tarea específica.

X = A/B

A = Utilización de memoria especificados en el diseño.

B = Utilización memoria solicitado en los requisitos.

Esta métrica no es evaluada.

VRRC = 0.

7.5.2.3 Densidad del mensaje de la utilización de la memoria.

El diseño debe contener cual es la densidad de mensajes referente a la utilización de entrada/salida en las líneas de código responsables de hacer las llamadas al sistema

X = A/B

A = Número de mensajes de error relacionado con la memoria especificadas en el diseño.

B = Número de líneas de código relacionadas directamente con las llamadas al sistema según los requisitos.

El diseño relaciona 10 mensajes de error relacionados a la memoria (A=10) y existen 10 líneas de código, una por cada aplicación, directamente relacionadas a las invocaciones del sistema (B=10). Por lo tanto, la "Densidad del mensaje de uso de la memoria" es FM = 1 (A/B).

Teniendo en cuenta el valor especificado por el cliente VRRC = 0.60, el resultado final es RM = 0.60

El diseño relaciona 20 mensajes de error relacionados a I/O (A=20). Existen 20 líneas de código, una para cada aplicación, que tienen como finalidad invocar el sistema correspondiente (B=20).

Por lo tanto, la "Densidad del mensaje de uso de I/O" es FM = 1 (A/B) Teniendo en cuenta el valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.40.

7.5.2.4 Densidad del mensaje de uso de entrada-salida.

El diseño debe contener cual es la densidad de mensajes de uso de entrada y salida.

X = A/B

A = Número de mensajes de error relacionado con el uso de entrada-salida especificados en el diseño.

B = Número de líneas de código relacionadas directamente con las llamadas al sistema según los requisitos.

El diseño relaciona 20 mensajes de error relacionados a I/O (A=20). Existen 20 líneas de código, una para cada aplicación, que tienen como finalidad invocar el sistema correspondiente (B=20).

Por lo tanto, la "Densidad del mensaje de uso de I/O" es FM = 1 (A/B) Teniendo en cuenta el valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.40.

7.5.2.5 Utilización de la capacidad de transmisión.

El diseño debe contener la capacidad de transmisión esperada.

X = A/B

A = Capacidad de transmisión especificada en el diseño.

B = Capacidad de transmisión especificada la cuál es designada para ser usada en los requisitos.

Esta métrica no es evaluada.

VRRC = 0.

7.5.3 Conformidad con la eficiencia.

Esta Subcaracterística no es evaluada, VRRC = 0.

7.5.3.1 Conformidad con la eficiencia.

Qué tan conforme es la eficiencia del diseño aplicado a las regulaciones, estándares y convenciones?

X = A/B

A = Número de datos correctamente ejecutados relacionados con la conformidad de la eficiencia confirmada en la evaluación.

B = Número total de la conformidad de los datos.

Esta Subcaracterística no es evaluada

VRRC = 0.

7.6 Mantenibilidad.

7.6.1 Capacidad de ser analizado.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Capacidad de ser analizado" es PPM = 0. De acuerdo al valor especificado por el cliente VRRC = 0.10, esta subcaracterística no se cumple de manera satisfactoria.

7.6.1.1 Preparación de la función de diagnóstico.

El diseño debe contener cómo llevar a cabo la disposición de las funciones de diagnóstico.

X = A/B

A = Número de funciones de diagnóstico ejecutadas especificadas en el diseño.

B = Número de funciones de diagnóstico especificadas en los requisitos.

En el diseño no hay funciones de diagnóstico (A=0) y una sola fue requerida (B=1). Por lo tanto, la "Preparación de la función de diagnóstico" es FM = 0 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.50, el resultado final es RM = 0.

7.6.2 Facilidad de cambio.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Facilidad de cambio" es PPM = 0.25. De acuerdo al valor especificado por el cliente VRRC = 0.20, esta subcaracterística se cumple de manera satisfactoria.

7.6.2.1 Facilidad de registrar los cambios.

Se registran adecuadamente los cambios a en el diseño?

X = A/B

A = Número de cambios confirmados en el diseño.

B = Total de funciones o módulos modificados.

El diseño tiene 4 funciones que sufrieron cambios, los cuales fueron confirmados en la revisión (A=4). El número total de funciones cambiadas es de 4 (B=4). Por lo tanto, la "Facilidad de registrar cambios" es FM = 1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.30, el resultado final es RM = 0.30.

7.6.2.2 Capacidad de control de cambio en el diseño.

Puede el usuario fácilmente identificar versiones del diseño?

Puede el mantenedor fácilmente cambiar el diseño y resolver problemas?

X = A/B

A = Número de cambios registrados en el diseño.

B = Número de cambios para rastrear los cambios del diseño.

El diseño tiene 8 cambios (A=8). El número total de cambios que deberían se registrados es de 8 (B=8). Por lo tanto, la "Capacidad de control de cambio en el diseño" es FM = 0.80 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.20.

7.6.3 Estabilidad.

Teniendo en cuenta los valores calculados en métricas, se determina que la "Estabilidad" es de PPM = 0.23. De acuerdo al valor especificado por el cliente VRRC = 0.10, esta subcaracterística se cumple de manera satisfactoria.

7.6.3.1 Localización del impacto de la modificación.

Qué tan grande es el impacto de la modificación sobre diseño?

X = A/B

A = Número de variables de datos afectadas por la modificación, confirmadas en la diseño.

B = Número total de variables.

Debido a los cambios y/o modificaciones, 4 datos se vieron afectados y fueron confirmados en la diseño (A= 4). Se cuenta con un total de 32 cabios (B=32). Por lo tanto, la "Localización del impacto de la modificación" es FM = 0,125 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.60, el resultado final es RM = 0.075.

7.6.3.2 Impacto del cambio.

Puede el usuario operar el software del sistema sin fallos después del mantenimiento?

Puede el mantenedor fácilmente mitigar fallos causados por efectos secundarios de mantenimiento?

X = 1 – (A/B)

A = Número impactos adversos en el diseño.

B = Número cambios realizados.

Luego de los cambios realizados (B= 4), no se detectaron impactos adversos en el diseño (A=0).

Por lo tanto, el "Impacto del cambio" es FM = 1 (1-A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.40.

7.6.4 Conformidad con la facilidad de mantenimiento.

Esta Subcaracterística no es evaluada, VRRC = 0

7.6.4.1 Conformidad con la facilidad de mantenimiento.

Qué tan conforme es la facilidad de mantenimiento del diseño a las regulaciones, a los estándares y a las convenciones aplicables?

X = A/B

A = Número de datos correctamente ejecutados relacionados con la conformidad de la facilidad de mantenimiento confirmadas en el diseño.

B = Número total de datos conformes.

Esta Subcaracterística no es evaluada.

VRRC = 0

7.7 Portabilidad.

7.7.1 Adaptabilidad.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Adaptabilidad" es de PPM = 0.20. De acuerdo al valor especificado por el cliente VRRC = 0.20, esta subcaracterística cumple de manera satisfactoria.

7.7.1.1 Adaptabilidad del diseño en las estructuras de datos.

Qué tan adaptable es diseño a los cambios de las estructuras de datos?

X = A/B

A = Número de estructuras de datos las cuales son operables y no tienen ninguna limitación después de la adaptación, confirmadas en el diseño.

B = Número total de estructuras de datos que requieren la capacidad de adaptación.

Este diseño tiene asociado una estructura de datos, la cual almacena la información procesada (A=1). Este tipo de estructura de datos requiere de la capacidad de adaptación (por ejemplo, agregar una nueva columna) (B=1). Por lo tanto, el "Adaptabilidad del diseño en las estructuras de datos" es FM =1 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.40, el resultado final es RM = 0.40.

7.7.1.2 Adaptabilidad del diseño en el ambiente de hardware.

Qué tan adaptable es el diseño en los cambios relacionados con el ambiente de hardware?

X = A/B

A = Número de funciones ejecutadas las cuales son capaces de alcanzar resultados requeridos en múltiples ambientes de Hardware según lo especificado, confirmadas en el diseño.

B = Número total de funciones con capacidad de requisitos de la adaptación.

Este diseño solo ejecuta las funciones implementadas en un solo ambiente de Hardware, es decir que las funciones implementadas no son capaces de lograr los resultados requeridos en varios ambientes de hardware (A=0). No existen funciones que puedan adaptarse a distintos ambientes de hardware (B=0). Por lo tanto, la "Adaptabilidad del diseño en el ambiente de hardware " es FM = 0 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.10, el resultado final es RM = 0.

7.7.1.3 Adaptabilidad del diseño en el ambiente organizacional.

Qué tan adaptable es el diseño al cambio organizacional?

X = A/B

A = Número de funciones ejecutadas las cuales son capaces de alcanzar resultados requeridos en múltiples ambientes de la organización y del negocio según lo especificado, confirmado en el diseño.

B = Número total de funciones con capacidad de requisitos de la adaptación del ambiente de la organización y del negocio.

Esta métrica no es evaluada.

VRRC = 0

7.7.1.4 Amigabilidad del diseño.

Puede el mantenedor adaptar fácilmente el diseño?

X = A/B

A = Número de funciones amigable en el diseño.

B = Número total de funciones que se adaptan fácilmente a los requerimientos.

En el diseño se determinó que 10 funciones son amigables al usuario (A=10). Dicha cantidad coincide con el número total de funciones que se adaptan fácilmente a los requerimientos (B=10). Por lo tanto, la "Amigabilidad del diseño" es FM = 1 (A/B). De acuerdo al valor especificado por el cliente VRRC = 0.20, el resultado final es RM = 0.20.

7.7.2 Reemplazabilidad.

Teniendo en cuenta los valores calculados en las métricas, se determina que la "Reemplazabilidad" es PPM = 0.45. De acuerdo al valor especificado por el cliente VRRC = 0.40, esta subcaracterística se cumple de manera satisfactoria.

7.7.2.1 Uso continuo de los datos.

Cuál es la cantidad de datos en el diseño original que siguen sin cambiar después de ser reemplazado?

X = A/B

A = Número de datos, que continúan siendo utilizados después del reemplazo, confirmado en el diseño.

B = Número total de datos que requieren ser usados desde los datos viejos después del reemplazo del diseño.

Luego de algunos reemplazos realizados, se tienen 38 items de datos (A=38). En la versión anterior de este diseño, se tenían 40 items de datos (B=40). Por lo tanto, el "Uso continuo de datos" es FM = 0.95 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.70, el resultado final es RM = 0.66.

7.7.2.2 Inclusividad de la función.

Cuál es la cantidad de funciones que siguen sin cambiar?

X = A/B

A = Número de funciones cubiertas por el nuevo diseño que produce resultados similares.

B = Número de funciones en el diseño anterior.

Luego de una revisión, se determinó que este diseño cuenta con 8 funciones que producen resultados similares (A=8). La versión anterior tenía 10 funciones (B=10). Por lo tanto, el "Inclusividad de la función" es FM = 0.80 (A/B). Teniendo en cuenta el valor especificado por el cliente VRRC = 0.30, el resultado final es RM = 0.24.

7.7.3 Conformidad con la portabilidad.

Esta Subcaracterística no es evaluada, VRRC = 0

7.7.3.1 Conformidad con la portabilidad.

Qué tan conforme es la portabilidad del diseño según las regulaciones, a los estándares y a las convenciones aplicables?

X = 1- A/B

A = Número de datos de la conformidad de la portabilidad que no han sido ejecutados durante la prueba.

B = Número total de datos especificados de la conformidad de la portabilidad.

Esta Subcaracterística no es evaluada.

VRRC = 0

7.8 Resultados por subcaracterísticas.

En la siguiente tabla se encuentra los resultados de la ejemplificación del modelo por subcaracterística.

También se encuentra el promedio ponderado de las métrica (PPM) el cual nos permite realizar la comparación versus el valor respectivo requerido por el cliente(VRRC) y poder concluir si la subcaracterística cumple o no cumple según el valores definidos por el cliente.

Funcionalidad	**Adecuación**	**Alcance de la implementación funcional**	**Valores**
		X = 1 - A/B	
		A	3
		B	10
		FM	0,7
		VRRC	0,2
		RM	0,14
		Estabilidad (o volatilidad) de la especificación funcional	**Valores**
		X = 1 - A/B	
		A	4
		B	10
		FM	0,6
		VRRC	0,3
		RM	0,18
		Suficiencia funcional	**Valores**
		X = 1 - A/B	
		A	3
		B	10
		FM	0,7
		VRRC	0,3
		RM	0,21

		Integridad de la implementación funcional	**Valores**
		X = 1 - A/B	
		A	0
		B	10
		FM	1
		VRRC	0,2
		RM	0,2
		Adecuación	**Valores**
		Alcance de la implementación funcional	**0,14**
		Estabilidad (o volatilidad) de la especificación funcional	**0,18**
		Suficiencia funcional	**0,21**
		Integridad de la implementación funcional	**0,2**
		PPM	**0,1825**
		VRRC	**0,3**
	Exactitud	**Exactitud en el diseño**	**Valores**
		X = A/B	
		A	7
		B	10
		FM	0,7
		VRRC	0,8
		RM	0,56
		Precisión	**Valores**

		X = A/B	
		A	10
		B	10
		FM	1
		VRRC	0,2
		RM	0,2
		Exactitud	**Valores**
		Exactitud en el diseño	**0,56**
		Precisión	**0,2**
		PPM	**0,38**
		VRRC	**0,3**
Interoperabilidad	**Compatibilidad de los datos**	**Intercambiabilidad de datos**	**Valores**
		X = A/B	
		A	1
		B	1
		FM	1
		VRRC	0,6
		RM	0,6
		Compatibilidad de los datos	**Valores**
		Intercambiabilidad de datos	**0,6**
		PPM	**0,6**
		VRRC	**0,1**
	con la inter	**Regulación de interoperabilidad**	**Valores**

		X = A/B	
		A	1
		B	1
		FM	1
		VRRC	0,4
		RM	0,4
		Conformidad con la interoperabilidad	**Valores**
		Regulación de interoperabilidad	**0,4**
		PPM	**0,4**
		VRRC	**0,1**
Fiabilidad	**Madurez**	**Detección de fallas**	**Valores**
		X = A/B	
		A	3
		B	10
		FM	0,3
		VRRC	0,4
		RM	0,12
		Remoción de fallos	**Valores**
		X = A/B	
		A	0
		B	3
		FM	0
		VRRC	0,4

		RM	0
		Madurez	**Valores**
		Detección de fallas	**0,12**
		Remoción de fallos	**0**
		PPM	**0,06**
		VRRC	**0,4**
	Tolerancia a fallos	**Prevención de fallas**	**Valores**
		X = A/B	
		A	3
		B	3
		FM	1
		VRRC	0,5
		RM	0,5
		Prevención de operación incorrecta	**Valores**
		X = A/B	
		A	10
		B	3
		FM	3,333333333
		VRRC	0,5
		RM	1,666666667
		Tolerancia a fallos	**Valores**
		Prevención de fallas	**0,5**
		Prevención de operación incorrecta	**1,666666667**

		PPM	**1,083333333**
		VRRC	**0,3**
	Conformidad	**Conformidad con la fiabilidad**	**Valores**
		VRRC	0
Usabilidad	**Apropiabilidad**	**Integridad de la descripción**	**Valores**
		X = A/B	
		A	10
		B	10
		FM	1
		VRRC	0,2
		RM	0,2
		Funciones Evidentes	**Valores**
		X = A/B	
		A	10
		B	10
		FM	1
		VRRC	0,2

		RM	0,2
		Comprensibilidad de la función	**Valores**
		X = A/B	
		A	2
		B	10
		FM	0,2
		VRRC	0,2
		RM	0,04
		Integridad de la descripción	**Valores**
		X = A/B	
		A	5
		B	10
		FM	0,5
		VRRC	0,3
		RM	0,15
		Apropiabilidad	**Valores**
		Integridad de la descripción	**0,2**
		Funciones Evidentes	**0,2**
		Comprensibilidad de la función	**0,04**
		Integridad de la descripción	**0,15**
		PPM	**0,1475**
		VRRC	**0,4**

	Conformidad de uso	**Conformidad con la usabilidad**	**Valores**
		VRRC	0
Eficiencia	**Comportamiento en el tiempo**	**Tiempo de respuesta**	**Valores**
		X = A/B	
		A	1
		B	3
		FM	0,333333333
		VRRC	0,1
		RM	0,033333333
		Tiempo del rendimiento de procesamiento	**Valores**
		VRRC	0
		Plazo de entrega	**Valores**
		X = A/B	
		A	6
		B	8
		FM	0,75
		VRRC	0,5
		RM	0,375

		Comportamiento en el tiempo	**Valores**
		Tiempo de respuesta	0,033333333
		Tiempo del rendimiento de procesamiento	0
		Plazo de entrega	0,375
		PPM	**0,204166667**
		VRRC	**0,3**
	Utilización de recursos	**Utilización de entrada/salida**	**Valores**
		VRRC	0
		Utilización de la memoria	**Valores**
		VRRC	0
		Densidad del mensaje de la utilización de la memoria	**Valores**
		X = A/B	
		A	10
		B	10
		FM	1
		VRRC	0,6
		RM	0,6
		Densidad del mensaje de uso de entrada-salida	**Valores**
		X = A/B	
		A	20
		B	20
		FM	1
		VRRC	0,4

		RM	0,4
		Utilización de la capacidad de transmisión	**Valores**
		VRRC	0
		Utilización de recursos	**Valores**
		Utilización de entrada/salida	0
		Utilización de la memoria	0
		Densidad del mensaje de la utilización de la memoria	0,6
		Densidad del mensaje de uso de entrada-salida	0,4
		Utilización de la capacidad de transmisión	0
		PPM	**0,5**
		VRRC	**0,3**
	Conformidad	**Conformidad con la eficiencia**	**Valores**
		VRRC	0
Mantenibilidad	**Capacidad de ser analizado**	**Preparación de la función de diagnóstico**	**Valores**
		X = A/B	
		A	0
		B	1
		FM	0

		VRRC	0,5
		RM	0
		Capacidad de ser analizado	**Valores**
		Preparación de la función de diagnóstico	**0**
		PPM	**0**
		VRRC	**0,1**
	Facilidad de cambio	**Facilidad de registrar los cambios**	**Valores**
		X = A/B	
		A	4
		B	4
		FM	1
		VRRC	0,3
		RM	0,3
		Capacidad de control de cambio en el diseño	**Valores**
		X = A/B	
		A	8
		B	8
		FM	1
		VRRC	0,2
		RM	0,2
		Facilidad de cambio	**Valores**
		Facilidad de registrar los cambios	**0,3**
		Capacidad de control de cambio en el diseño	**0,2**

		PPM	**0,25**
		VRRC	**0,2**
	Estabilidad	**Localización del impacto de la modificación**	**Valores**
		X = A/B	
		A	4
		B	32
		FM	0,125
		VRRC	0,6
		RM	0,075
		Impacto del cambio	**Valores**
		X =1- (A/B)	
		A	0
		B	4
		FM	1
		VRRC	0,4
		RM	0,4
		Estabilidad	**Valores**
		Localización del impacto de la modificación	**0,075**
		Impacto del cambio	**0,4**
		PPM	**0,2375**
		VRRC	**0,1**

	Conformidad	**Conformidad con la facilidad de mantenimiento**	**Valores**
		VRRC	0
Portabilidad	**Adaptabilidad**	**Adaptabilidad del diseño en las estructuras de datos**	**Valores**
		X = A/B	
		A	1
		B	1
		FM	1
		VRRC	0,4
		RM	0,4
		Adaptabilidad del diseño en el ambiente de hardware	**Valores**
		X = A/B	
		A	0
		B	0
		FM	0
		VRRC	0,1
		RM	0
		Adaptabilidad del diseño en el ambiente organizacional	**Valores**

		VRRC	0
		Amigabilidad del diseño para el usuario	**Valores**
		X = A/B	
		A	10
		B	10
		FM	1
		VRRC	0,2
		RM	0,2
		Adaptabilidad	**Valores**
		Adaptabilidad del diseño en las estructuras de datos	**0,4**
		Adaptabilidad del diseño en el ambiente de hardware	**0**
		Adaptabilidad del diseño en el ambiente organizacional	**0**
		Amigabilidad del diseño para el usuario	**0,2**
		PPM	**0,2**
		VRRC	**0,2**
	Reemplazabilidad	**Uso continuo de los datos**	**Valores**
		X = A/B	
		A	38
		B	40
		FM	0,95
		VRRC	0,7
		RM	0,665

		Inclusividad de la función	**Valores**
		X = A/B	
		A	8
		B	10
		FM	0,8
		VRRC	0,3
		RM	0,24
		Reemplazabilidad	**Valores**
		Uso continuo de los datos	**0,665**
		Inclusividad de la función	**0,24**
		PPM	**0,4525**
		VRRC	**0,4**
	Conformidad	**Conformidad con la portabilidad**	**Valores**
		VRRC	0

Tabla 37 Resultados de los valores de las métricas .

7.9 Análisis de resultados.

Acorde al valor respectivo requerido por el cliente (VRRC) y los resultados obtenidos por cada subcaracterística para el promedio ponderado de las métrica (PPM) se puede concluir si cada una de las subcaracterísticas cumple o no cumple.

En la siguiente tabla se recolecta los resultados de cada subcaracterística de manera que se pueda identificar de manera simple cuales de estas cumplieron, no cumplieron o no es requiera a tener en cuenta por el cliente.

Característica	Subcaracterística	Cumple	No Cumple	No Requerida
Funcionalidad	Adecuación		1	
	Exactitud	1		
Interoperabilidad	Compatibilidad de los datos	1		
	Conformidad con la interoperabilidad	1		
Fiabilidad	Madurez		1	
	Tolerancia a fallos	1		
	Conformidad con la fiabilidad			1
Usabilidad	Apropiabilidad		1	
	Conformidad de uso			1
Eficiencia	Comportamiento en el tiempo		1	
	Utilización de recursos	1		
	Conformidad con la eficiencia			1
Mantenibilidad	Capacidad de ser analizado		1	
	Facilidad de cambio	1		

	Estabilidad	1		
	Conformidad con la facilidad de mantenimiento			1
Portabilidad	Adaptabilidad	1		
	Reemplazabilidad	1		
	Conformidad con la portabilidad			1
Total		**9**	**5**	**5**

Tabla 38 Análisis de resultados por subcaracterística .

La tabla a continuación resumen para cada una de las características cuantas subcaracterísticas se cumplieron, no se cumplieron o no fueron requeridas por el cliente.

Característica	**Cumple**	**No Cumple**	**No Requerida**	**Total**
Funcionalidad	1	1	0	**2**
Interoperabilidad	2	0	0	**2**
Fiabilidad	1	1	1	**3**
Usabilidad	0	1	1	**2**
Eficiencia	1	1	1	**3**
Mantenibilidad	2	1	1	**4**
Portabilidad	2	0	1	**3**
Total	**9**	**5**	**5**	**19**

Tabla 39 Análisis de resultados por característica.

De un total de 24 subcaracterísticas, existen 5 subcaracterísticas que deberán ser consideradas. Esto representa un 53% de cumplimiento y un 47% de no cumplimiento. De esta forma, se deberá revisar y/o mejorar las métricas asociadas a las subcaracterísticas.

Según la siguiente tabla se tiene que con un 53% hay un grado insatisfactorio.

Escala de Medición	**Nivel de Puntuación**	**Grado de Satisfacción**
0%-40%	Inaceptable	Insatisfactorio
40,1% - 70%	Mínima Aceptable	
70,1% - 90%	Rango Objetivo	Satisfactorio
90,1% - 100%	Excede los Requisitos	

Tabla 40 Valores para los criterios de aceptación.

En la siguiente tabla se encuentra un resumen por cada subcaracterística que permite analizas de acuerdo al PPM si este cumple de acurdo el VRRC definido por el cliente.

Característica	**Subcaracterística**	**VRRC Valores Respectivos Requeridos por el Cliente**	**PPM Promedio Ponderado de sus Métricas**
Funcionalidad	Adecuación	0,3	0,1825
	Exactitud	0,3	0,38
Interoperabilidad	Compatibilidad de los datos	0,1	0,6
	Conformidad con la	0,1	0,4

	interoperabilidad		
Fiabilidad	Madurez	0,4	0,06
	Tolerancia a fallos	0,3	1,083333333
	Conformidad con la fiabilidad	N/A	N/A
Usabilidad	Apropiabilidad	0,4	0,1475
	Conformidad de uso	N/A	N/A
Eficiencia	Comportamiento en el tiempo	0,3	0,204166667
	Utilización de recursos	0,3	0,5
	Conformidad con la eficiencia	N/A	N/A
Mantenibilidad	Capacidad de ser analizado	0,1	0
	Facilidad de cambio	0,2	0,25
	Estabilidad	0,1	0,2375
	Conformidad con la facilidad de mantenimiento	N/A	N/A
Portabilidad	Adaptabilidad	0,2	0,2
	Reemplazabilidad	0,4	0,4525
	Conformidad con la portabilidad	N/A	N/A

Tabla 41 Valores requeridos por el cliente y promedio ponderado de las métricas. .

El siguiente grafico muestra si el PPM esta cumpliendo con el VRRC para cada una de las subcaracterísticas.

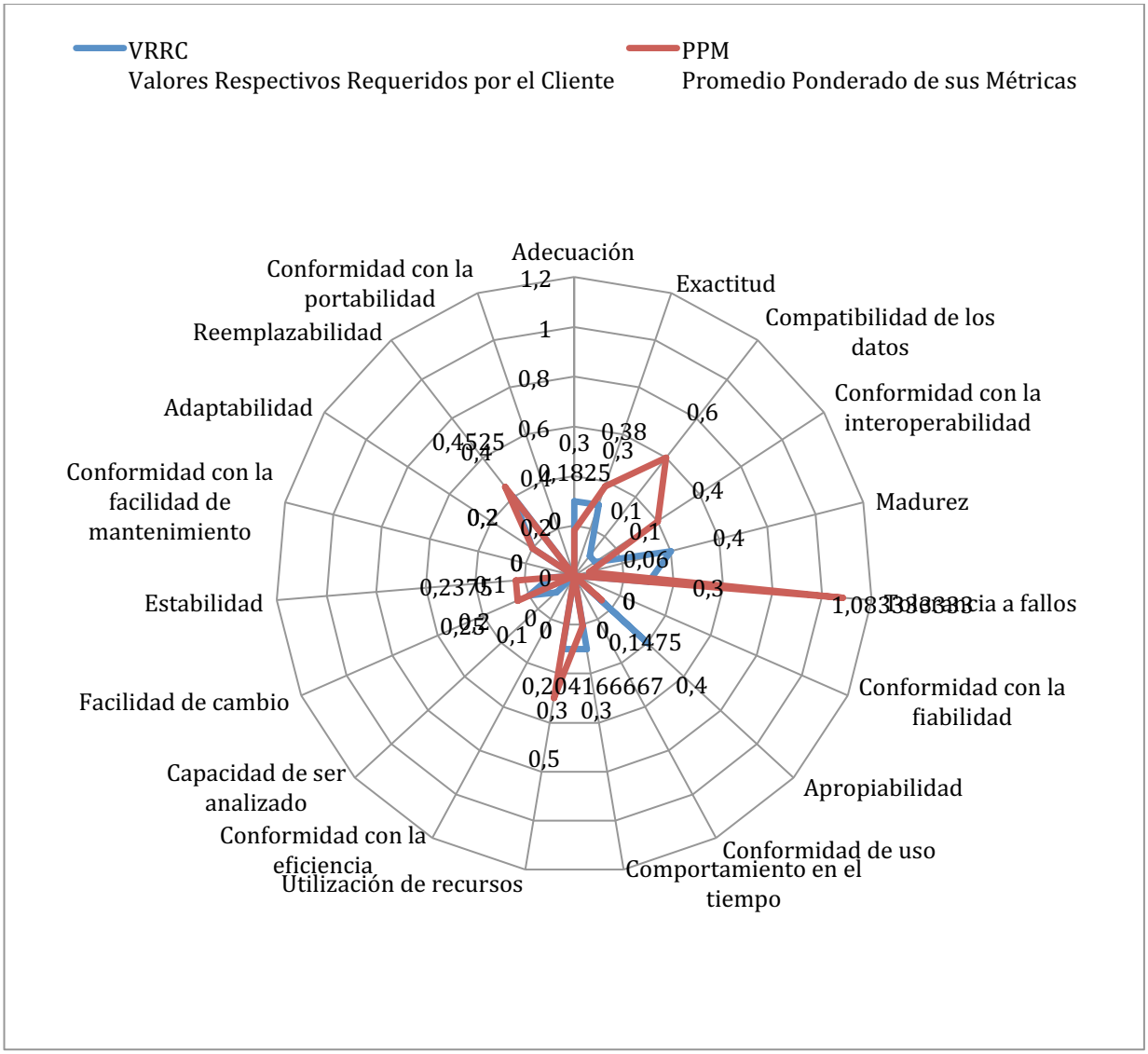

Imagen 17: VRRC VS PPM.

Se adjunta la ejemplificación del modelo con todos los cálculos realizados y la grafica de radar.

8 Conclusiones y recomendaciones.

8.1 Conclusiones.

- A pesar que el modelo no se aplico a un caso de la vida real, la ejemplificación que se realizo de este permite determinar que tiene validez.
- La aplicación del modelo puede aportar conclusiones importantes que permiten determinar si el diseño esta cumpliendo con los niveles mínimos para que este sea aceptado.
- Se deben definir de manera clara los valores respectivo requeridos por el cliente debido a que estos serán los que permitirán evaluar si el diseño cumple con los niveles mínimos de calidad.
- Es de vital importancia definir los VRRC acorde a las necesidades de calidad.
- El modelo no tiene en cuenta todas las características y subcaracterísticas contenidas en la familia de normas ISO/IEC 25000.
- El modelo aporta cerrando la brecha los re-procesos en la etapa de desarrollo.
- ISO/IEC 25000 por estar relacionada directamente con el tema de calidad es parte fundamental para el aseguramiento de la calidad en el diseño del software en la metodología elaborada en el presente proyecto.
- Teniendo en cuenta la debilidad existente en cuanto a estándares informáticos y normatividades dirigidas exclusivamente al diseño de software, la metodología planteada en este proyecto propone una solución pertinente para el aseguramiento de la calidad en el diseño del software.

- La metodología PHVA (Planear, Hacer, Verificar y Actuar), cumplen un papel vital en la aplicación de la metodología debido a que son estas quienes determinan un aseguramiento en la cálida exitosa.

8.2 Recomendaciones.

- El aseguramiento en la calidad en el diseño del software debe considerarse como una etapa antes de iniciar la construcción y después de la elicitación de los requisitos del mismo, permitiendo que las fases siguientes de en el ciclo de vida se ejecuten con el mínimo riesgo de pérdida de calidad en el producto informático a entregar.
- Es sumamente importante que el modelo se apliquen con rigurosidad y de manera completa para garantizar la fiabilidad en el aseguramiento en la calidad del diseño.

9 Bibliografía.

Grady, R.B. (1990): "Work-product analysis: the philosopher's stone of software?", **IEEE Software**, March, pag.26-34.

David Garlan and Mary Shaw: "An introduction to Software Architecture".

PRESSMAN Roger S., 2005. *Ingeniería de Software. Un enfoque práctico.* Sexta edición. 2005, Estados Unidos**.** McGrawHill.

Jacobson, I., P. Jonsson, M. Christerson and G. Overgaard, Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.

Zavala, J. 2004. ¿Por qué fracasan los proyectos de software? Un enfoque organizacional.

Piattini , García, "Calidad en el desarrollo y mantenimiento del software", RA-MA Editorial, Madrid, 2003.

International Organization for Standarization. Software Engineering – Product Quality. ISO/IEC 9126. Geneva, Suiza: ISO, 2006. P.612.

International Organization for Standarization. Software Engineering – Systems and software Quality Requerements and Evaluation (SQuaRE) – System and software quality models. ISO/IEC FDIS 25010. Geneva, Suiza: ISO 2010. P.34.

Olsina, L., Covella, G. y Rossi, G. (2005). Web Quality. Capítulo del libro Web Engineering: Theory and Practice of Metrics and Measurement for Web Development. Emilia Mendes y Nile Mosley Editores.A publicarse por Spinger Verlag en 2005. ISBN: 3-540-28196-7.

Polillo R. (2011). Quality Models for Web 2.0 Sites: a Methodological Approach and a Proposal.2nd Workshop on The Web and Requirements Engineering (WeRE'11)In (ICWE 2011).

Portal ISO 25000. Calidad del Producto Software. En: http://iso25000.com.

Resolución 61/2005. Creación del registro nacional de productores de software y servicios informáticos. Procedimiento de inscripción. Consulta: 08 de enero del 2013. Disponible en: http://www.infoleg.gov.ar/infolegInternet/anexos/105000-109999/106061/norma.htm.

http://www.blog-top.com/el-ciclo-phva-planear-hacer-verificar-actuar/

Pressman, R. 2005. Ingenieria del Software. 6ª Ed. Mcgraw-Hill. Parte III, cap. 16-19.
Sommerville, I. (2005), Ingeniería del Software. 7ª Edición. Ed. Pearson.

ISO/IEC. ISO/IEC 25000:2005 Software engineering - Software product QUAlity Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. Technical report, International Organization for Standardization, Geneva, Switzerland, 2005.

ISO/IEC. ISO/IEC 9126-1:2001, Software engineering - Product quality - Part 1: Quality model. Technical report, International Organization for Standardization, Geneva, Switzerland, 2001.

ISO/IEC. ISO/IEC 14598-1:1999, Information technology - Software product evaluation - part 1: General overview. Technical report, International Organization for Standardization, Geneva, Switzerland, 1999.

ISO/IEC. ISO/IEC 15939:2002 - Software Engineering – Software Measurement Process. Technical report, International Organization for Standardization, Geneva, Switzerland, 2002.

Witold Suryn, Alain Abran, and Alain April. ISO/IEC square. the second generation of standards for software product quality. Technical report, 2002.

R.G. Dromey Software Quality Institute Griffith University Nathan, Brisbane, QLD 4111 AUSTRALIA. A MODEL FOR SOFTWARE PRODUCT QUALITY.

Luis E. Mendoza, Maria A. Perez y Anna C. Grimán. Universidad Simon Bolívar. Departamento de procesos y sistemas. Prototipo de modelo sistemico de calidad (MOSCA) del Software.

Julia González Rodríguez, Luis Olsina. Departamento de informática, Escuela Politécnica, UnEx. Hacia la medición de calidad en uso web.

Henry,S. y Selig, C. (1990): "Predicting source code complexity at the design stage", IEEE Software, March, pag.36-45.

Rombach, H.D. (1990): "Design measurement: some lessons learned", **IEEE Software**, March , pag.17-25.

Juan P. Carvallo, Xavier Franch, Carme Quer, Xavier Burgués, Gemma Grau. Universitat Politècnica de Catalunya (UPC). COSTUME: Un método para la combinación

Grady, R.B. (1990): "Work-product analysis: the philosopher's stone of software?", **IEEE Software**, March, pag.26-34.

Piedrahita, Sebastián; Construcción de una herramienta para evaluar la calidad de un producto software; Proyecto de Grado, Departamento de Informática y Sistemas, Universidad EAFIT, 2007.