Universidad de las Ciencias Informáticas

Facultad 3

Título: Guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Adrian Fernández Alvarez

Adrian Pompa Peña

Tutor: Ing. Raúl Velázquez Alvarez.

Ciudad de la Habana

Junio 2012

"Podrán morir las personas, pero jamás sus ideas."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

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

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

_____	_____	_____
Adrian Fernández Alvarez	Adrian Pompa Peña	Ing. Raúl Velázquez Alvarez
Autor	Autor	Tutor

DATOS DE CONTACTO

Ingeniero en Ciencias Informáticas
Centro de trabajo: Universidad de las Ciencias Informáticas.
Categoría docente: Instructor
Cargo que desempeña: Asesor Técnico Docente de Calidad del Centro de Gobierno Electrónico de la facultad 3 y líder del Proyecto Calidad del mismo centro.
Años de experiencia: 3
Correo electrónico: rvelazquez@uci.cu

AGRADECIMIENTOS

A mis familiares y amigos que estuvieron presentes en todo momento a lo largo de toda mi vida de estudiante dándome fuerzas y apoyo en todo momento en estos años de estudio, gracias por estar siempre conmigo y por ser mi guía a seguir en la vida, los quiero y los admiro mucho.

A mi compañero de tesis por ser más que un amigo, un hermano.

A los profesores sin excepción de ninguno, que durante estos años han hecho de mí una persona más capacitada y preparada para enfrentar los grandes retos que han de venir en lo adelante como profesional.

A nuestro tutor y a todos los que de una forma u otra hicieron posible la realización de nuestra tesis.

Adrian Pompa Peña

AGRADECIMIENTOS

Quisiera darle las gracias a mi mama, porque en todo momento me apoyo y me dio el ánimo suficiente para salir adelante en todas las situaciones que se me presentaron durante estos 5 años, muchas gracias por estar siempre ahí para mí cuando lo necesite. Me enseñaste que la vida es más que sacrificio y te doy mil gracias, porque si no es por ti, nunca hubiera llegado este momento, el primer ingeniero de la FAMILIA.

Quisiera darle las gracias a mi papa, porque siempre me ayudo y estuvo pendiente de mí, eres un padre ejemplar, siempre oigo tus consejos y si alguien en mi familia admiro es a ti. Muchas gracias por estar siempre ahí para mí y que momentos tan lindos como este se sigan repitiendo.

Quisiera agradecer a mi compañero de tesis Adrian Pompa que verdaderamente sin él esta tesis no se hubiera realizado y siempre le agradeceré lo que hizo por mí en todo este tiempo de Universidad, es un gran amigo y espero que nunca perdamos el contacto, ni la amistad.

Quiero agradecer a mi novia Arick por tener que soportarme durante todo este tiempo y darme su apoyo incondicional cuando siempre me hizo falta.

Quisiera agradecer a varias personas que siempre han estado ahí para mí cuando me ha hecho falta alguna cosa, mi chicas lindas Anaelis y Ariadna, Leinis que verdaderamente te agradezco lo que hiciste por mí en los primeros años de la Universidad, a mi hermano Fabián que con su locura siempre le ha dado un toque de alegría a mi vida y a todas aquellas personas que de una forma u otra me ayudaron y apoyaron.

A mi tutor Raúl Velázquez por habernos apoyado con nuestra investigación en todo momento.

A mi grupo ya que compartimos todo este tiempo de universidad como una gran familia, ojala que la amistad que tenemos nunca se pierda.

Darle las gracias a mi gente del Futbol, que todas las tardes compartimos gratos momentos.

Adrian Fernández Alvarez

DEDICATORIA

A mi familia por su amor, a mis amigos por su apoyo y a la Revolución Cubana por haberme dado la oportunidad de ser ingeniero.

Adrian Pompa Peña

Les dedico esta tesis a mis padres y a mi familia, a toda la gente que me apoyo y a la Revolución Cubana por haberme dado la oportunidad de ser ingeniero.

Adrian Fernández Alvarez

RESUMEN

La mantenibilidad en los sistemas informáticos viene dada por la cantidad de modificaciones o cambios realizados al software. Uno de los principales problemas que influyen en el software actualmente está relacionado con esta característica, al no implementarla con la calidad requerida en los productos informáticos se hace más complejo corregir las modificaciones, ya que a mayor mantenibilidad menor costo de mantenimiento y viceversa. La Universidad de las Ciencias Informáticas fue creada con el propósito de utilizar los recursos computacionales de tal manera que se produzcan soluciones eficientes y eficaces a los problemas informáticos, contando con una red de centros para la creación de estos productos. Entre ellos se encuentra el Centro de Gobierno Electrónico (CEGEL), el cual no cuenta con el personal capacitado o con los conocimientos necesarios para garantizar la mantenibilidad en sus productos. Este trabajo está enfocado en el estudio de la mantenibilidad como característica de calidad, sus principales conceptos, costes y problemas, además de profundizar en los principales aspectos para planificar, gestionar y ejecutar eficientemente dicho proceso y proponer una guía para asegurar su calidad durante el proceso de desarrollo en los sistemas de Gobierno Electrónico. Una vez confeccionada la guía con los elementos a tener en cuenta por cada etapa del ciclo de vida del software, se somete a una evaluación según el criterio de un grupo de especialistas en el tema, y por último se emplea el coeficiente de Kendall para medir el nivel de la concordancia de los especialistas.

PALABRAS CLAVE

Aseguramiento, Calidad, Guía, Mantenibilidad.

TABLA DE CONTENIDOS

INTRODUCCIÓN

A finales de los 60 ya se había popularizado el término de Crisis del Software debido a los constantes cambios en el desarrollo del software y hardware, ya que no se contaba con las metodologías, herramientas y modelos para gestionar los productos, problema que ha repercutido hasta la actualidad; para darle solución a esta crisis surge un área en la informática que recibe el nombre de Ingeniería del Software, la cual brinda un conjunto de principios y métodos con el objetivo de obtener un software de modo rentable y fiable. Una de las principales causas de esta situación ha sido la poca importancia que se le ha atribuido a la característica de calidad mantenibilidad, la cual supone alrededor del 80% de los costes del ciclo de vida de un producto de software.

La mantenibilidad es la capacidad que tiene el producto para ser modificado, estas modificaciones pueden ser mejoras, nuevos requerimientos, corrección de errores, así como la petición y gestión de respuestas. La mantenibilidad del software es una de las características más difíciles de lograr, por esto es importante tener modelos y/o estándares para poder controlarla e implementarla con la calidad requerida en el producto que se está desarrollando, ya que sin el correcto uso de estos modelos al realizar las correcciones de los errores a menudo se realizan sin pensarse o con precipitación y debido a esto no son documentadas adecuadamente o introducidas correctamente con el código existente. Al no contar con un conocimiento para realizar este proceso trae consigo la introducción de nuevos errores e ineficiencias, y con esto un incremento en los esfuerzos y los recursos dedicados a estos.

La Universidad de las Ciencias Informáticas (UCI) tiene como meta convertirse en una institución donde sus productos de software tengan la calidad requerida para los clientes, contando con un personal capacitado para que obtengan el máximo beneficio o por lo menos un beneficio aceptable en un período de tiempo establecido; para lograr esta meta se encuentra el Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos (Calisoft), encargado de propiciar los modelos y estándares internacionales para medir cada categoría de los procesos y con esto poder controlar la evolución de la calidad.

Según un estudio realizado a integrantes de proyectos como arquitectos, líderes de proyectos, desarrolladores, analistas de la Universidad de las Ciencias Informáticas (UCI) pertenecientes al Centro de Gobierno Electrónico (CEGEL) el cual es encargado del desarrollo de soluciones jurídicas, se arribó a que algunos proyectos como Informatización de los Tribunales Populares Cubanos (TPC), Sistema de Gestión Fiscal (SIGEF), Sistema de Información Registral de la Cámara de Comercio (SIRECC), entre otros, emplean algunas metodologías de desarrollo de software como RUP, AUP y XP que les permiten estructurar, planificar y controlar el proceso de desarrollo, no cuentan con el personal capacitado ni la

experiencia necesaria para garantizar estos aspectos relacionados con la mantenibilidad, debido a esto cuando se realizan modificaciones que pueden ser correcciones, adaptación del software a cambios en el entorno, en los requerimientos o en las especificaciones, esto incurre en un mayor tiempo dedicado al proceso de mantenibilidad, en un ineficiente uso de los recursos, así como la introducción de nuevos errores y un incremento en los costos.

A partir de la situación planteada anteriormente se define como **problema a resolver**:

¿Cómo contribuir al aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico de manera que estos reduzcan el tiempo dedicado al proceso de mantenibilidad y el correcto uso de los recursos utilizados?

De ahí que el **objeto de estudio** de la investigación sea: aseguramiento de la calidad.

Como **objetivo general**: proponer una guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico de manera que estos reduzcan el tiempo dedicado al proceso de mantenibilidad y el correcto uso de los recursos.

Y como **campo de acción:** aseguramiento de la mantenibilidad de los sistemas informáticos de Gobierno Electrónico.

Se plantea como **idea a defender** que: proponiendo una guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico se contribuye a que estos reduzcan el tiempo dedicado al proceso de mantenibilidad y el correcto uso de los recursos utilizados.

El objetivo se desglosa en una serie de **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Desarrollar la propuesta de solución.
- ✓ Validar la solución propuesta.

Se plantea una serie de **tareas de la investigación** para dar cumplimiento a los objetivos de la investigación:

- ✓ Realización de un estudio del estado del arte de la temática en Cuba y el mundo.

- ✓ Análisis de la Norma (ISO/IEC 25010) División Modelo de Calidad y otros estándares internacionales que traten el tema de las características de calidad de software.
- ✓ Análisis de las sub-características existentes en dependencia del estudio de los estándares y modelos definidos internacionalmente que traten el tema de las características de calidad de software.
- ✓ Análisis de los diferentes atributos existentes que conforman cada sub-característica.
- ✓ Definición de los atributos necesarios durante el ciclo de desarrollo, basado en el programa de mejora de la UCI que plantea CMMI, para que las aplicaciones web y desktop de Gobierno Electrónico cumplan con la característica de Mantenibilidad.
- ✓ Definición de acciones o actividades a realizar a lo largo del ciclo de desarrollo de software, para asegurar la Mantenibilidad en el producto final, basándose en los elementos anteriormente definidos.
- ✓ Validación de la propuesta de solución.

A continuación se relacionan los **métodos de investigación** utilizados:

Métodos Teóricos

- Analítico – Sintético: permite analizar las teorías y documentos permitiendo la extracción de los elementos más importantes que se relacionan con el tema a desarrollar.
- Histórico – Lógico: se utiliza para realizar un estudio sobre la evolución del atributo de calidad mantenibilidad y el desarrollo de sus diferentes sub características.

Métodos Empíricos

- Entrevistas: facilita obtener información relacionada con el atributo de calidad mantenibilidad en los proyectos de CEGEL, la cual fue aplicada a los integrantes de los proyectos del centro.
- Encuesta: se utiliza para realizar un cuestionario a un grupo de especialistas con el objetivo de validar la investigación mediante sus criterios.

La estructura de la investigación es la siguiente: consta de introducción, 3 capítulos, conclusiones, recomendaciones, referencias bibliográficas, anexos y glosario de términos.

Capítulo 1: En este capítulo se hace un análisis del estado del arte de los diferentes modelos y/o estándares de calidad relacionados con la mantenibilidad como: McCall, ISO 9126 y ISO/IEC 25010. Además se presentan algunos conceptos como: calidad de software, aseguramiento de la calidad, mantenibilidad y de las subcaracterísticas relacionadas con esta última.

Capítulo 2: En este capítulo se describe el ciclo de vida por el cual se deben regir los proyectos de la universidad, se definen pautas a tener en cuenta por cada etapa de este ciclo de vida y finalmente se presenta la guía propuesta.

Capítulo 3: En este capítulo se realiza la validación de la propuesta a partir de la aplicación del criterio de especialistas, se presentan los resultados de un cuestionario después de haber sido aplicado a personas con experiencia profesional en el tema y que tienen el conocimiento necesario para emitir su criterio.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se presentan las principales definiciones relacionados con la calidad del software, aseguramiento de la mantenibilidad, la mantenibilidad y sus subcaracterísticas, los principales aspectos que influyen en la mantenibilidad y los diferentes modelos y/o estándares de calidad utilizados para el desarrollo de esta investigación.

1.2 Calidad de software

El significado de esta palabra puede adquirir múltiples interpretaciones, ya que todo dependerá del nivel de satisfacción o conformidad del cliente. Sin embargo, la calidad es el resultado de un esfuerzo arduo, se trabaja de forma eficaz para poder satisfacer el deseo del consumidor. Dependiendo de la forma en que un producto o servicio sea aceptado o rechazado por los clientes, se puede decir si este es bueno o malo.

Según la Real Academia Española se define como la: *"Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor".*

Según la ISO 9000: *"Grado en el que un conjunto de características inherentes cumple con los requisitos".*

Según Philip B. Crosby: *"Calidad es cumplimiento de requisitos".*

Según William E. Deming: *"Calidad es satisfacción del cliente".*

Según la ISO 25000: *"Grado en que el producto de software satisface las necesidades expresadas o implícitas, cuando es usado bajo condiciones determinadas".*

Según la IEEE, Std 610-1900: *"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario".*

Según Pressman, 2010: "*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".*

Según lo analizado anteriormente, los autores definieron un propio concepto para el desarrollo de este trabajo: *"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y establecidos, con los estándares de desarrollo prefijados y las necesidades o expectativas que desea el cliente o usuario".*

El objetivo no es necesariamente alcanzar una calidad perfecta, sino la necesaria y suficiente en las etapas de desarrollo, luego trazando estrategias y actividades para lograr el aseguramiento y control de la calidad.

1.3 Aseguramiento de la mantenibilidad

El aseguramiento de calidad se enfoca desde tempranas etapas en el desarrollo del producto para identificar y evaluar los defectos que puedan afectar al software. Si los errores se pueden identificar de forma temprana en el proceso de software, las características del diseño de software se pueden especificar de modo que eliminarán o controlarán los peligros potenciales. Debido a esto es necesario conocer algunos conceptos que definen el aseguramiento de la calidad para llegar a una definición sobre el aseguramiento de la mantenibilidad:

El aseguramiento de calidad del software es el "*conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad."* (Cueva Lovelle , 1999)

(Sridhar, 2000) indica que *"mientras el software que se está desarrollando reúne los requerimientos y su desempeño sea el esperado, es preciso que se supervisen las actividades de desarrollo del software y su rendimiento, en distintas oportunidades durante cada fase del ciclo de vida. Este es el papel del aseguramiento de la calidad del software".*

Según (Fuertes Castro, 2008) es *"Conjunto de actividades de planificación, estimación y supervisión del proceso de desarrollo, que se realizan de forma independiente al equipo de desarrollo, de tal forma que los productos software resultantes cumplen los requisitos establecidos y satisfacen los niveles de calidad exigidos."*

Según lo analizado anteriormente, los autores definieron un propio concepto para el desarrollo de este trabajo: *"El aseguramiento de la mantenibilidad del software es un diseño planificado de acciones o actividades a realizar a lo largo del ciclo de vida de software que se requieren para garantizar la Mantenibilidad en el producto final."*

Logrando una correcta supervisión del proceso de desarrollo, así como las actividades de planificación y estimación permite que las modificaciones a realizar en el producto de software sean más fáciles de corregir y controlar.

1.4 Mantenibilidad

El IEEE (19990) define mantenibilidad como*: "La facilidad con la que un sistema o componente de software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno".*

Esta definición está directamente conectada con la definición del IEEE para mantenimiento del software: *"es el proceso de modificar un componente o sistema software después de su entrega para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarlo a cambios en el entorno".*

Según la ISO/IEC 25010 la define como: "*la capacidad del producto de software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a los cambios en el medio ambiente, en los requisitos y especificaciones funcionales".*
Este estándar define unas series de sub-características para el proceso de mantenibilidad:

- ✓ Analizabilidad: la capacidad del producto de software para ser diagnosticado por deficiencias o causas de fallas en el software.

- ✓ Cambiabilidad: la capacidad del producto de software para permitir una modificación especificada a implementar.

- ✓ Estabilidad: la capacidad del producto de software para evitar efectos inesperados de las modificaciones del software.

- ✓ Capacidad de prueba: la capacidad del producto de software para permitir que el software modificado sea probado.

- ✓ Facilidad de mantenibilidad: la capacidad del producto de software para adherirse a normas o convenciones relacionadas con el mantenimiento.

1.4.1 Propiedades de la mantenibilidad

La mantenibilidad se puede considerar como la combinación de dos propiedades diferentes: reparable y la flexible.

- ✓ Reparable: Un software es reparable si permite la corrección de sus defectos con una cantidad de trabajo limitado y razonable. Esta se ve afectada por la cantidad y tamaño de los componentes o módulos. Un producto de software que consiste en módulos bien diseñados es más fácil de analizar y reparar que uno monolítico, pero el incremento del número de estos no implica un producto más reparable, ya que también aumenta la complejidad de las interconexiones entre ellos. Así pues, se debe buscar un punto de equilibrio de la estructura entre estos componentes más adecuada, para garantizar que sea reparable facilitando la localización y eliminación de los errores en unos pocos módulos.

- ✓ Flexible: Un software es flexible si permite cambios para que se satisfagan nuevos requerimientos, es decir, si puede evolucionar. Por su naturaleza inmaterial, el software es mucho más fácil de cambiar o incrementar por lo que respecta a sus funciones que otros productos de naturaleza física, por ejemplo, equipos hardware, pero esta flexibilidad se ve disminuida con cada nueva versión de un producto de software, ya que cada versión complica la estructura del software y, por tanto, las futuras modificaciones serán más difíciles. Aunque esto puede considerarse una generalidad, la aplicación de técnicas y metodologías apropiadas pueden minimizar el impacto en la flexibilidad de cada nueva modificación en el software. Es por esto que la flexibilidad es una característica tanto del producto de software como de los procesos relacionados con su construcción. En términos de estos últimos, los procesos deben poderse acomodar a nuevas técnicas de gestión y organización, a cambios en la forma de entender la ingeniería, etc.

1.5 Modelos y/o estándares internacionales de calidad a nivel de producto.

Con la constante evolución de las tecnologías en la actualidad es necesario contar con modelos y/o estándares para poder desarrollar y medir la calidad de los productos, para lograr las necesidades de los usuarios finales.

Según la ISO/IEC: *"Un estándar es un documento publicado. Se trata de un conjunto de reglas que se usan de manera habitual como los buenos principios, prácticas o directrices. Un estándar controla cómo las personas deben desarrollar y gestionar los materiales, productos, servicios, tecnologías, procesos y sistemas."*

1.5.1 Modelo de McCall (1977)

El modelo de McCall fue el primero en ser presentado en 1977, se enfoca en el producto final, identificando atributos claves desde el punto de vista del usuario. Estos atributos se denominan factores de calidad y son normalmente atributos externos, pero también se incluyen algunos atributos posiblemente internos.

Los factores de calidad son demasiados abstractos para ser medidos directamente, por lo que por cada uno de ellos se introducen atributos de bajo nivel denominados criterios de calidad. Algunos criterios de calidad son atributos internos, reflejando la creencia de McCall que el atributo interno tiene un efecto directo en el atributo externo correspondiente.

Un nivel más de descomposición es necesario, mapeando cada criterio de calidad en un conjunto de métricas de calidad que son atributos (tanto del producto como del proceso) de muy bajo nivel, medibles directamente.

Propone tres perspectivas para agrupar los factores de Calidad:

- Revisión del producto: habilidad para ser cambiado, en la que se incluyen los siguientes factores de calidad.
  - ✓ Mantenibilidad: esfuerzo requerido para localizar y corregir fallas.
  - ✓ Flexibilidad: facilidad de realizar cambios.
  - ✓ Testeabilidad: facilidad para realizar las pruebas, para asegurarse que el producto no tiene errores y cumple con la especificación.

- Transición del producto: adaptabilidad al nuevo ambiente, en la que se incluyen los siguientes factores de calidad.

- ✓ Portabilidad: esfuerzo requerido para transferir entre distintos ambientes de operación.
- ✓ Reusabilidad: facilidad de reusar el software en diferentes contextos.
- ✓ Interoperabilidad: esfuerzo requerido para acoplar el producto con otros sistemas.

- Operación del producto: características de operación, en la que se incluyen los siguientes factores de calidad:
  - ✓ Correctitud: el grado en el que el producto cumple con su especificación.
  - ✓ Confiabilidad: la habilidad del producto de responder ante situaciones no esperadas.
  - ✓ Eficiencia: el uso de los recursos tales como tiempo de ejecución y memoria de ejecución.
  - ✓ Integridad: protección del programa y sus datos de accesos no autorizados.
  - ✓ Usabilidad: facilidad de operación del producto por parte de los usuarios.

El factor mantenibilidad incluye los siguientes criterios: consistencia, simplicidad, concisidad, auto-descripción, modularidad. Pero la mantenibilidad ha cambiado bastante desde 1977; encontrar y corregir errores es sólo un aspecto más.

La medición de cualquiera de estos factores está definida en este modelo en base a 41 métricas, para cada criterio existe una lista de condiciones que se deben cumplir en distintas etapas: requerimientos, diseño e implementación. Se cuentan las condiciones que se satisfacen en cada una de las etapas sobre el total posible, eso da una medida del criterio, que se pondera en partes iguales para medir el factor con los otros criterios asociados al factor.

El modelo McCall tiene sus límites, entre los que se encuentran:

- ✓ Es difícil que las características y subcaracterísticas sean siempre perfectamente independientes.
- ✓ Falta una asociación explícita entre los modelos y el proceso de software, así como realizar software de calidad.
- ✓ Las características son en general propiedades abstractas medibles mediante métricas. No siempre existe una relación perfectamente lineal entre los valores de las métricas y las características que deben estimar.

1.5.2 ISO 9126

Durante muchos años se buscó en la Ingeniería de Software un modelo único para expresar calidad, la ventaja era obvia: poder comparar productos entre sí. En 1992, una variante del modelo de McCall fue propuesta como estándar internacional para medición de calidad de software la ISO 9126 Evaluación del Producto de Software: Características de Calidad y Guía para su uso (del inglés Software Product Evaluation: Quality Characteristics and Guidelines for their Use), es el nombre formal de la última revisión.
El modelo de calidad que presenta este estándar cambia durante el ciclo de vida:

- ✓ Al principio, durante la recopilación de requerimientos y análisis, la calidad es especificada por los requisitos del usuario, sobre todo desde el punto de vista externo.
- ✓ En la fase de diseño e implementación, la calidad externa se traduce en un diseño técnico, confrontándose con el punto de vista de los desarrolladores sobre la calidad interna y complementándose con los requisitos implícitos que el software debe cumplir.
- ✓ La calidad final (la del uso) debe ser apropiada para los usuarios y el contexto de uso no existe una calidad perfecta o absoluta. Existe solamente una calidad necesaria y suficiente para un dado contexto.

En ISO 9126 se reconocen seis factores de calidad que se pueden considerar tanto internos como externos como se muestra en la figura 1.1:

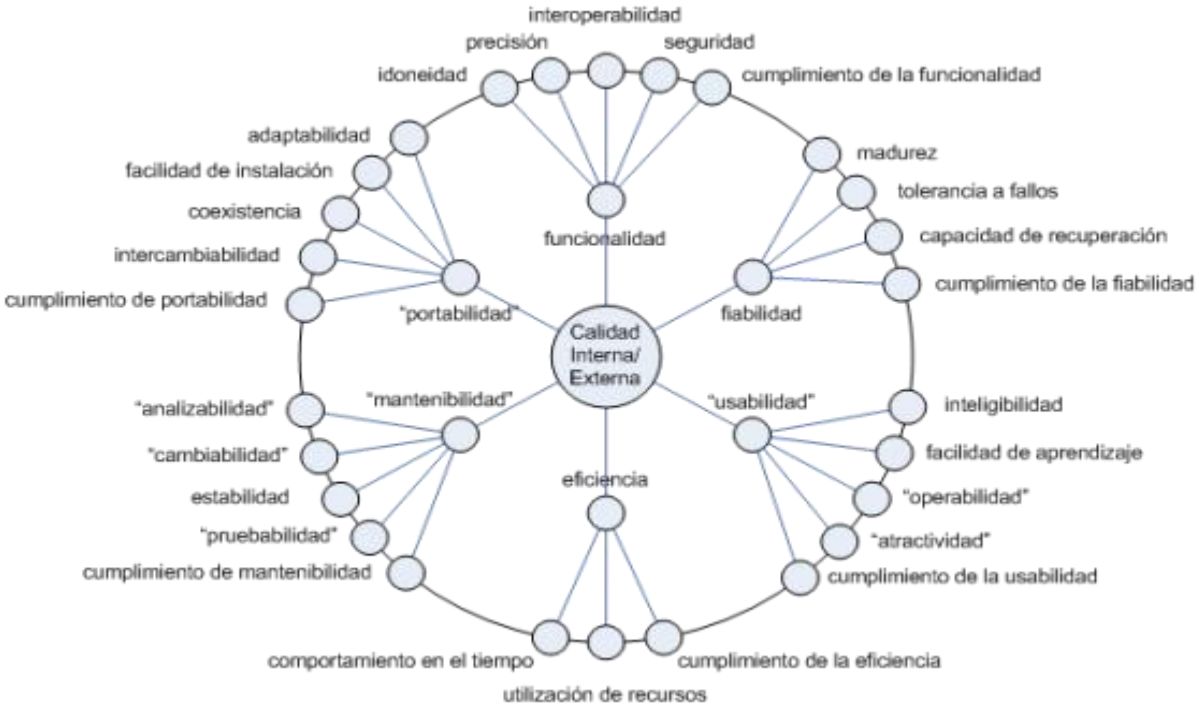

Figura 1.1 Modelo de calidad para la calidad externa e interna ISO 9126.

La mantenibilidad se define como el conjunto de atributos que se relacionan con el esfuerzo en realizar modificaciones y las subcaracterísticas que presenta son:

- ✓ Analizabilidad: atributos que miden el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallas, o para identificación de las partes que deben ser modificadas.
- ✓ Facilidad para el cambio: atributos que miden el esfuerzo necesario para realizar modificaciones, corrección de fallas o cambios en el contexto.
- ✓ Estabilidad: atributos que se relacionan con el riesgo de efectos no esperados en las modificaciones.
- ✓ Testeabilidad: atributos que miden el esfuerzo necesario para validar el software modificado.
- ✓ Cumplimiento: atributos que hacen que el software se adhiera a estándares relacionados con la aplicación y convenciones o regulaciones legales.

ISO 9126 provee 3 conjuntos de métricas, para medir respectivamente las características externas (en ISO 9126-2), las internas (en ISO 9126-3), y las de uso (en ISO 9126-4). Entre las que se encuentran para medir la mantenibilidad: Analizabilidad, Cambiabilidad, Estabilidad, Capacidad de prueba, entre otras.

El objetivo del uso de las métricas tanto internas como externas son la de representar la calidad de un producto de software respecto a las características y subcaracterísticas del modelo 9126, durante el testeo validar el cumplimento del software respecto a los requisitos de calidad externa, predecir el nivel de calidad de uso del producto, describir el grado de respuesta del producto respecto a los requisitos explícitos e implícitos de su uso. Representar la calidad de un producto de software en los estados de evolución intermedios y finales no ejecutables, respecto a las características y subcaracterísticas del modelo 9126, predecir el nivel de calidad externo del producto, prevenir problemas en el uso del producto, descubriendo anticipadamente potenciales defectos, las métricas internas son en general combinación de métricas elementales aplicadas a código fuente, diagramas UML, gráficos, etc. (medidas mediante análisis estático o con inspección de código). En el mantenimiento se podrá evaluar el esfuerzo y el riesgo de modificar un programa usando métricas de mantenibilidad.

La norma ISO 9126 ofrece una mejor organización comparada con McCall, se presenta de una forma sencilla de cómo definir la calidad de los sistemas, ya que está pensado para los desarrolladores,

adquirentes, personal que asegure la calidad y evaluadores independientes, responsables de especificar y evaluar el producto de software. Por tanto, puede servir para validar la completitud de una definición de requisitos, identificar requisitos de software, objetivos de diseño y prueba, criterios de aseguramiento de la calidad, etc. La calidad de cualquier proceso del ciclo de vida del software influye en la del producto de software que a su vez, contribuye a mejorarla en el uso del producto. Este estándar está en proceso de revisión, el modelo de calidad del software se incluirá en el sistema de normas ISO 25000 en la norma ISO 2501n, sin modificaciones.

1.5.3 ISO/IEC 25010

El modelo SQuaRE: Evaluación de los Requisitos de Calidad de los Productos de Software (del inglés Software Product Quality Requirements and Evaluation) es una revisión de ISO/IEC 9126, este modelo conserva 6 características de calidad de software y le agrega 2 nuevas las que se muestran en la figura 1.2. La serie SQuaRE tiene las siguientes divisiones:

- ISO/IEC 2500n. División de dirección de calidad.
- ISO/IEC 2501n. División del modelo de calidad.
- ISO/IEC 2502n. División de medida de calidad.
- ISO/IEC 2503n. División de requisitos de calidad.
- ISO/IEC 2504n. División de evaluación de calidad.

Se centrará en las divisiones 2501n y 2502n, en la primera se presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso. En la segunda incluyen un modelo de referencia de calidad del producto de software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Entre las métricas internas relacionadas con los atributos de mantenibilidad se encuentran:

- ✓ Capacidad de ser analizado: registro de la actividad, preparación de la función de diagnóstico.
- ✓ Facilidad de cambio: facilidad de registrar los cambios.
- ✓ Estabilidad: impacto en el cambio, localización del impacto de la modificación.
- ✓ Facilidad de prueba: integridad de la función de prueba predefinida, autonomía de la facilidad de prueba, facilidad de observación del desarrollo de prueba.
- ✓ Conformidad con la facilidad de mantenimiento: conformidad con la facilidad de mantenimiento.

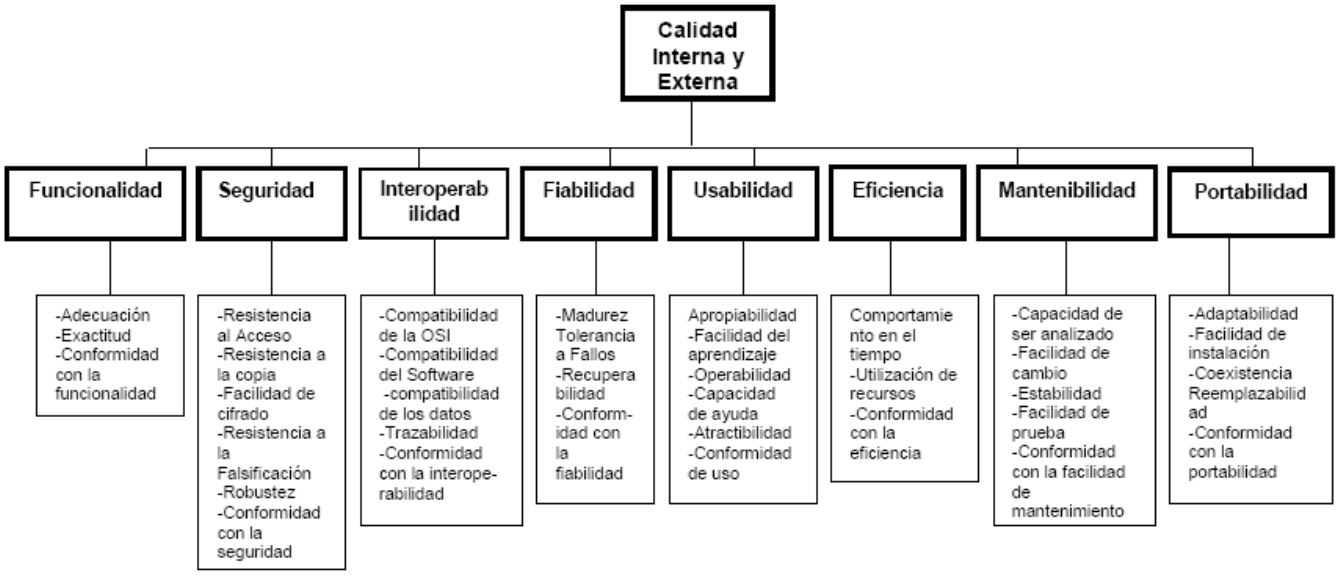

Figura 1.2 Modelo de calidad para la calidad externa e interna en ISO/IEC 2501n.

Los beneficios de utilizar SQuaRE son:

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

Por lo tanto se decidió escoger el modelo SQuaRE para la realización de esta investigación ya que es la siguiente generación a la ISO 9126, en la que se establecen criterios para la especificación de requisitos de calidad de productos de software, sus métricas y su evaluación, e incluye un modelo de calidad para

unificar las definiciones de calidad de los clientes con los atributos en el proceso de desarrollo, intentando definir lo más preciso posible cada una de las características y subcaracterísticas, demostrando así que existe más precisión en esta y es hasta ahora es el más actualizado y adecuado para el desarrollo de esta guía.

1.6 Aspectos que influyen en la mantenibilidad

La mantenibilidad debe establecerse como objetivo tanto en las etapas iniciales del ciclo de vida, para reducir las posteriores necesidades de mantenimiento, como durante las etapas de mantenimiento, para reducir los efectos laterales y otros inconvenientes ocultos (y seguir así reduciendo las futuras necesidades de mantenimiento).

Existen unos pocos factores que afectan directamente a la mantenibilidad, de forma que si alguno de ellos no se satisface adecuadamente, esta se resiente. Los tres más significativos son:

- ✓ Proceso de desarrollo: la mantenibilidad debe formar parte integral del proceso de desarrollo del software. Las técnicas utilizadas deben ser lo menos intrusivas posible con el software existente. Los problemas que surgen en muchas organizaciones de mantenimiento son de doble naturaleza: mejorar la mantenibilidad y convencer a los responsables de que la mayor ganancia se obtendrá únicamente cuando la mantenibilidad esté incorporada intrínsecamente en los productos software.
- ✓ Documentación: En múltiples ocasiones, ni la documentación ni las especificaciones de diseño están disponibles, y por tanto, los costes del mantenimiento se incrementan debido al tiempo requerido para que el encargado del mantenimiento entienda el diseño del software antes de poder ponerse a modificarlo. Las decisiones sobre la documentación que debe desarrollarse son muy importantes cuando la responsabilidad del mantenimiento de un sistema se va a transferir a una organización nueva.
- ✓ Comprensión de Programas: La causa básica de la mayor parte de los altos costes del mantenimiento es la presencia de obstáculos a la comprensión humana de los programas y sistemas existentes. Estos obstáculos surgen de tres fuentes principales: la información disponible es incomprensible, incorrecta o insuficiente. La complejidad del software, de la naturaleza de la aplicación o de ambos. La confusión, mala interpretación u olvidos sobre el programa o sistema.

Dependiendo de cómo se haya construido el software se puede aumentar la mantenibilidad. Los generadores de código, por lo general, no producen un código claro ni fácil de comprender, por lo que el mantenimiento del software así generado es peor. Por otro lado, las técnicas de programación estructurada, la aplicación de metodologías de ingeniería del software y el seguimiento de estándares, permiten la obtención de sistemas o componentes software con menos necesidades de mantenimiento, y en el caso de que se produzcan, mucho más fáciles de llevar a cabo.

Más concretamente, se han identificado algunos de los factores que influyen en la mantenibilidad:

- ✓ Falta de cuidado en las etapas de diseño, codificación o prueba.
- ✓ Pobre configuración del producto de software.
- ✓ Adecuada cualificación del equipo de desarrolladores del software.
- ✓ Estructura del software fácil de comprender.
- ✓ Facilidad de uso del sistema.
- ✓ Empleo de lenguajes de programación y sistemas operativos estandarizados.
- ✓ Estructura estandarizada de la documentación.
- ✓ Documentación disponible de los casos de prueba.
- ✓ Incorporación en el sistema de facilidades de depuración.
- ✓ Disponibilidad del equipo (computador y periféricos) adecuado para realizar el mantenimiento.
- ✓ Disponibilidad de la persona o grupo que desarrolló originalmente el software.
- ✓ Planificación del mantenimiento.

Es necesario tener en cuenta estos factores que influyen en la mantenibilidad para hacer el sistema más mantenible y a su vez, facilitar el proceso de mantenimiento. Es necesario analizar los principales problemas que afectan al mantenimiento para reunir todos los elementos necesarios para lograr que el producto sea más fácil de mantener, a continuación se hace mención a ellos:

Código heredado: con el paso de los años se ha ido produciendo un volumen muy grande de software. En la actualidad, la mayor parte de este software está formado por código antiguo “heredado” (del inglés legacy code); es decir, código de aplicaciones desarrolladas hace algún tiempo, con técnicas y herramientas en desuso y probablemente por personas que ya no pertenecen al colectivo responsable en este momento del mantenimiento del software. En muchas ocasiones la situación se complica porque el

código heredado fue objeto de múltiples actividades de mantenimiento. La opción de desechar este software y reescribirlo para adaptarlo a las nuevas necesidades tecnológicas o a los cambios en la especificación es muchas veces inadecuada por la gran carga financiera que supuso el desarrollo del software original y la necesidad económica de su amortización.

Los problemas específicos del mantenimiento de código heredado han sido caracterizados en las llamadas Leyes del Mantenimiento del Software [Lehman et al, 1980]:

- ✓ Continuidad del Cambio: un programa utilizado en un entorno del mundo real debe cambiar, ya que si no cada vez será menos usado en dicho entorno. En otras palabras, esto significa que, tan pronto como un programa ha sido escrito, está ya desfasado. Las razones que conducen a esta afirmación son varias: a los usuarios se les ocurren nuevas funcionalidades cuando comienzan a utilizar el software; nuevas características en el hardware pueden permitir mejoras en el software; se encuentran defectos en el software que deben ser corregidos; el software debe instalarse en otro sistema operativo o máquina; o el software necesita ser más eficiente.

- ✓ Incremento de la Complejidad: a la par que los cambios transforman los programas, su estructura se hará progresivamente más compleja salvo que se haga un esfuerzo activo para evitar este fenómeno. Esto significa que al realizar cambios en un programa, la estructura de dicho programa se hace más compleja cuando los programadores no pueden o no quieren usar técnicas de ingeniería del software.

- ✓ Evolución del Programa: la evolución de un programa es un proceso autorregulado. Las medidas de determinadas propiedades como el tamaño, tiempo entre versiones y número de errores revelan estadísticamente determinadas tendencias.

- ✓ Conservación de la Estabilidad Organizacional: a lo largo del tiempo de vida de un programa, la carga que supone el desarrollo de dicho programa es aproximadamente constante e independiente de los recursos dedicados.

- ✓ Conservación de la Familiaridad: durante todo el tiempo de vida de un sistema, el incremento en el número de cambios incluidos con cada versión es aproximadamente constante.

Casi veinte años después, el mismo autor vuelve a confirmar las leyes anteriores [Lehman et al., 1998]: la afirmación *"los grandes programas no llegan nunca a completarse y están en constante evolución"* se ve confirmada por el hecho de que, como se ha mencionado, algo más del 60% de las modificaciones que se realizan en el software se refieren al proceso de mantenimiento.

Además de las dificultades de mantenimiento que señalan las leyes anteriores, existen otros problemas clásicos que complican el mantenimiento [Schneidewind, 1987]:

- ✓ A menudo, el mantenimiento es realizado en un estilo libre establecido por el propio programador (esta opinión también es compartida por Pressman [2010], quien afirma *que "raramente existen organizaciones formales, de modo que el mantenimiento se lleva a cabo como se pueda"*). No en todas las ocasiones esta situación es debido a la falta de tiempo para producir una modificación diseñada cuidadosamente. Prácticamente todas las metodologías se han centrado en el desarrollo de nuevos sistemas y no han tenido en cuenta la importancia del mantenimiento. Por esta razón, no existen o son poco conocidos los métodos, técnicas y herramientas que proporcionan una solución global al problema del mantenimiento.

- ✓ Cambio tras cambio, los programas tienden a ser menos estructurados. Esto se manifiesta en una documentación desfasada (como afirman Baxter y Pigdeon [1997] al indicar que *"la documentación completa o inexistente del sistema es uno de los cuatro problemas más importantes del mantenimiento de software")*, código que no cumple los estándares, incremento en el tiempo que los programadores necesitan para entender y comprender los programas o en el incremento en los efectos secundarios producidos por los cambios. Todas estas situaciones implican casi siempre unos costes de mantenimiento del software muy alto.

- ✓ Es muy habitual que los sistemas que están siendo sometidos a mantenimiento sean cada vez más difíciles de cambiar (lo cual, como confirman Griswold y Notkin [1993], provoca que el mantenimiento sea cada vez más costoso). Esto se debe al hecho de que los cambios en un

programa por actividades de mantenimiento dificultan la posterior comprensión de la funcionalidad del programa. Sommerville [1992] también apunta que *"cualquier cambio conlleva la corrupción de la estructura del software y, a mayor corrupción, la estructura del programa se torna menos comprensible y más difícil de modificar"*. Por ejemplo, el programa original puede basarse en decisiones de programación no documentadas a las que no puede acceder el personal de mantenimiento. En estas situaciones, es normal que el software no pueda ser cambiado sin correr el riesgo de introducir efectos colaterales no deseados debido a interdependencias entre variables y procedimientos que el personal de mantenimiento no ha detectado.

- ✓ La falta de una metodología adecuada suele conducir a que los usuarios participen poco durante el desarrollo del sistema software. Esto tiene como consecuencia que, cuando el producto se entrega a los usuarios, no satisface sus necesidades y se tienen que producir esfuerzos de mantenimiento mayores en el futuro.

- ✓ También pueden existir problemas de gestión. Muchos programadores consideran el trabajo de mantenimiento como una actividad inferior menos creativa que les distrae del trabajo del desarrollo de software. Esta visión puede verse reforzada por las condiciones laborales, salariales y esto crea una baja moral entre las personas dedicadas al mantenimiento. Como resultado de lo anterior, cuando se hace necesario realizar mantenimiento, en vez de emplear una estrategia sistemática, las correcciones tienden a ser realizadas con precipitación o sin pensarse, no son documentadas adecuadamente y pobremente integradas con el código existente. No es extraño, pues, que el propio mantenimiento conduzca a la introducción de nuevos errores e ineficiencias que conducen a nuevos esfuerzos de mantenimiento con posterioridad.

Todos estos problemas se pueden atribuir parcialmente al gran número de programas existentes que han sido desarrollados sin tener en cuenta la Ingeniería del Software. Es por ello que es necesario conocer los principales problemas del mantenimiento para no incurrir en estos y controlarlos desde tempranas etapas para hacer el producto más mantenible así como sus efectos secundarios.

Efectos secundarios del mantenimiento

La posibilidad de error al cambiar un procedimiento lógico tan complejo como el que constituyen la mayor parte de los programas actuales es muy grande. Por esta razón, una de las principales dificultades del mantenimiento del software es el riesgo del llamado efecto bola de nieve de manera que los cambios producidos por una petición de mantenimiento introducen efectos secundarios que implicarán nuevas peticiones de mantenimiento en el futuro. Estos efectos secundarios suponen nuevos defectos que aparecen como consecuencia de las modificaciones realizadas.

Según las consecuencias que se derivan, los efectos secundarios del mantenimiento del software son de tres clases [Freedman y Weinberg, 1990]:

- ✓ Efectos secundarios sobre el código: todos los desarrolladores de software han sufrido en algún momento de su vida profesional los problemas originados por olvidar añadir un (;) o por confundir por un simple error de mecanografía un signo de puntuación con otro. Las consecuencias de estos despistes pueden ser muy importantes y sirven para corroborar que los efectos secundarios por cambios en el código son difíciles de prever. Las modificaciones en el código fuente que tienen una mayor probabilidad de inducir a nuevos errores son: cambios en el diseño que suponen muchos cambios en el código, eliminación o modificación de un subprograma, eliminación o modificación de una etiqueta, eliminación o modificación de un identificador, cambios para mejorar el rendimiento, modificación de la apertura/cierre de ficheros o modificación de operaciones lógicas.

- ✓ Efectos secundarios sobre los datos: las estructuras de datos constituyen una parte fundamental y básica en cualquier producto de software, por lo que cualquier cambio que se produzca en ellas puede conducir a fallos importantes del sistema. Los efectos secundarios de este tipo pueden aparecer debido a los siguientes cambios: redefinición de constantes locales o globales, modificación de los formatos de registros o archivos, cambio en el tamaño de una matriz u otras estructuras similares, modificación de la definición de variables globales, reinicialización de indicadores de control o punteros, o cambios en los argumentos de los subprogramas.

  Para reducir esta clase de efectos secundarios es importante una correcta documentación de todos los datos, incluyendo tablas de referencias cruzadas que los asocien con los subprogramas que los utilizan.

- ✓ Efectos secundarios sobre la documentación: los efectos secundarios de esta clase se producen cuando los cambios sobre el código de una aplicación no se reflejan en la documentación de diseño y/o en la documentación de usuario. Si la documentación técnica no se corresponde con el estado actual del software, se producirán efectos secundarios debidos a una incorrecta caracterización de las propiedades de dicho software. Por otro lado, la estima que los usuarios tendrán del producto de software se reducirá considerablemente si comprueban que la documentación no se adapta a los ejecutables. Los cambios que con mayor probabilidad pueden producir efectos secundarios sobre la documentación son: modificar el formato de las entradas interactivas, nuevos mensajes de error no documentados, tablas o índices no actualizados o textos no actualizados correctamente. Es muy recomendable revisar la configuración entera del software, incluyendo la documentación, para evitar estos efectos secundarios. De hecho, existen peticiones de mantenimiento que se pueden satisfacer sólo con corregir, ampliar o clarificar la documentación sin necesidad de producir cambios en los programas.

1.7 Modelo Integrado de Capacidad y Madurez (CMMI).

CMMI dirige su enfoque a la mejora de procesos en una organización, estudia los procesos de desarrollo y produce una evaluación de la madurez de la organización. Los modelos contienen los elementos esenciales de procesos efectivos para una o más disciplinas y describen el camino para evolucionar y mejorar desde procesos inmaduros a procesos disciplinados, maduros con calidad, eficiencia mejorada y probada. Se establecen 5 niveles de madurez para clasificar a las organizaciones. Para cumplir con un determinado nivel es necesario cumplir con todas las actividades definidas para ese nivel y para los niveles anteriores.

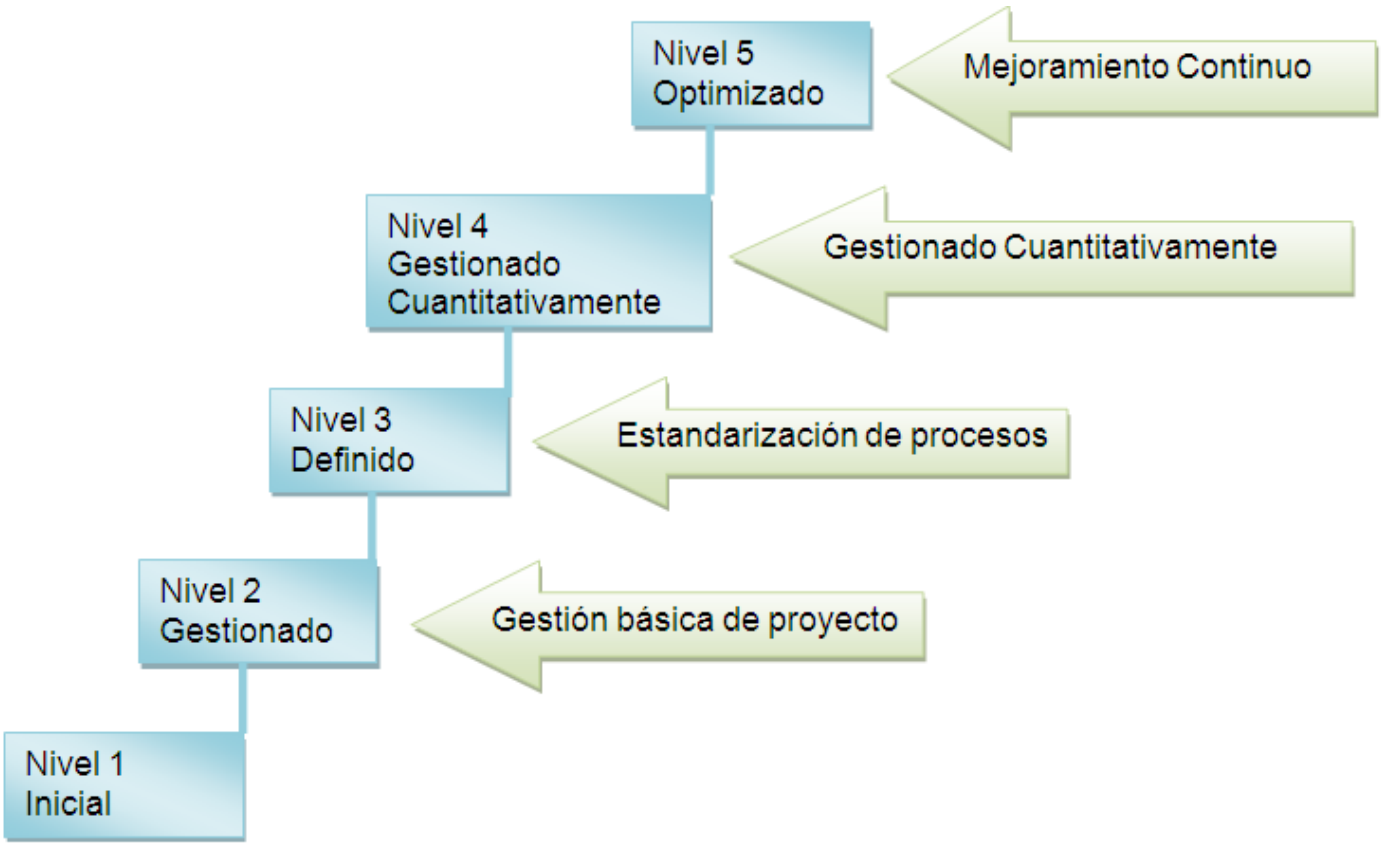

Figura 1.3 Representación Escalonada

En el nivel 2 cuenta con 7 áreas de procesos, que son: Administración de Acuerdo con Proveedores (SAM), Aseguramiento de la Calidad de Procesos y Productos (PPQA), Administración de la Configuración (CM), Planeación del Proyecto (PP), Monitoreo y Control de Proyecto (PMC), Administración de Requisitos (REQM) y Medición y Análisis (MA).

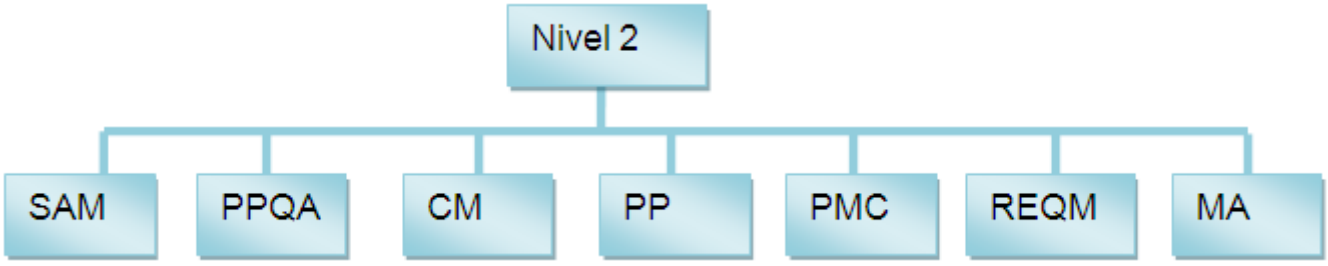

Figura 1.4 Áreas de procesos del nivel 2

Con el nivel 2 de CMMI se pretende conseguir que en los proyectos de la organización haya una correcta gestión de los requisitos y que los procesos estén planeados, ejecutados, medidos y controlados.

Para la realización de este trabajo se escogió el ciclo de vida que plantea el programa de mejora en el nivel 2 de CMMI, ya que es por el que se rigue actualmente la universidad.

Conclusiones parciales

En el presente capítulo se realizó un estudio de los principales conceptos de calidad del software, aseguramiento de la calidad, modelos y/o estándares internacionales asociados al proceso de mantenibilidad, así como los principales aspectos que influyen en esta característica, los cuales servirán de soporte para el desarrollo de la propuesta de solución y definir los elementos a tener en cuenta para asegurar la calidad de esta característica en los productos del Centro de Gobierno Electrónico (CEGEL).

Después de analizar los estándares de calidad expuestos, se realizó una comparación entre estos y se decidió escoger el modelo SQuaRE ya que reúne todas las características necesarias para la realización de esta investigación, así como un conjunto de métricas para medir la calidad de los atributos que posee la mantenibilidad.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo se presenta la guía para asegurar la mantenibilidad en sistemas informáticos de Gobierno Electrónico, utilizando el ciclo de vida de los proyectos definidos en el nivel 2 de CMMI propuestos en el programa de mejora que se lleva a cabo en la universidad. Se describen en cada una de las etapas las actividades a realizar que atentan contra esta característica.

2.2 Características de los proyectos del centro CEGEL

El centro de Gobierno Electrónico está compuesto por dos departamentos: el departamento de Informática Jurídica (IJ) y el departamento de Gestión Gubernamental (GG), en los cuales se desarrollan sistemas informáticos utilizando plataformas como Java, PHP y Microsoft .NET entre otros. Los sistemas que se desarrollan en este centro se caracterizan por tener una gran cantidad de información, procesos complejos, un personal poco organizado y estos u otros aspectos atentan contra la mantenibilidad del software. Estas soluciones se rigen por el ciclo de vida definido por el programa de mejora que se lleva a cabo en la universidad y se definió un conjunto de actividades o elementos a tener en cuenta que influyen para lograr asegurar la mantenibilidad a lo largo del ciclo de desarrollo del producto.

2.3 Ciclo de vida del proceso de desarrollo de software basado en el programa de mejora

Como parte de la institucionalización que lleva a cabo la universidad es necesario realizar un estudio del ciclo de vida del proceso de desarrollo de software y utilizarlo como punto de partida para definir por cada una de las etapas del mismo, las actividades que se deben tener en cuenta para el aseguramiento de la mantenibilidad. (Calisoft, 2011)

Ciclo de Vida
Estudio Preliminar
Modelación del Negocio
Requisitos
Análisis y Diseño
Implementación
Pruebas Internas
Pruebas de Liberación

Soporte

Tabla 2.1 Ciclo de Vida

2.3.1 Estudio preliminar

En la etapa de Estudio Preliminar se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel. En esta etapa se realiza un estudio inicial de la organización del cliente que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

Elementos a tener en cuenta:

- ✓ Impartir capacitaciones.

2.3.2 Modelación del negocio

El Modelado del Negocio es la etapa destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para la descripción y modelado de negocio pueden ser utilizadas diferentes técnicas como el Modelado de Casos de Uso del Negocio y Notación de Modelado de Procesos del Negocio (del inglés Business Process Modeling Notation).

Elementos a tener en cuenta:

- ✓ Reglas del negocio.

2.3.3 Requisitos

El esfuerzo principal en la etapa de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de casos de uso, servicios que describen todas las interacciones que tendrán los usuarios con el software, estos responden a los requisitos funcionales del sistema. Además la especificación de requisitos incluye requisitos no funcionales.

Elementos a tener en cuenta:

- ✓ Especificación de Requisitos de Software.

2.3.4 Análisis y Diseño

Durante esta etapa, de considerarse necesario, a través de los Modelos de Análisis los requisitos descritos durante la etapa de Requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de los mismos y una descripción que sea fácil de mantener y ayude a estructurar el sistema (incluyendo su arquitectura).

Durante esta etapa es modelado el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la implementación. Los modelos desarrollados en esta etapa son más formales y específicos de una implementación. Durante esta etapa son desarrollados el documento de arquitectura, diagramas de clases, diagramas de entidad relación, diagrama de despliegue entre otros.

Elementos a tener en cuenta:

- ✓ Elección de patrones de diseño.
- ✓ Elección de estilos arquitectónicos.
- ✓ Exactitud y organización lógica de los datos.

2.3.5 Implementación

En la implementación a partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares.

Elementos a tener en cuenta:

- ✓ Disponer de la documentación generada y artefactos relacionados con la implementación.
- ✓ Garantizar la legibilidad del código.
- ✓ Uso de las buenas prácticas de programación.
- ✓ Detección de errores en el código.

2.3.6 Pruebas internas

Durante esta etapa el proyecto verifica el resultado de la implementación probando según sea necesaria cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Durante esta etapa se deben desarrollar artefactos de prueba como: Diseños de casos de prueba, Listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.

Elementos a tener en cuenta:

- ✓ Disponer de toda la documentación generada y artefactos.

- ✓ Selección de métodos de evaluación arquitectónicos.

2.3.7 Pruebas de liberación

Pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

No se definen elementos ya que esta prueba es realizada por una entidad ajena a la organización, ellos definirán los elementos a tener en cuenta.

2.3.8 Despliegue

Durante esta etapa se procede a la entrega de la solución, así como a la instalación, configuración, prueba y puesta en marcha del software en el entorno real del cliente. Las pruebas de esta etapa incluyen pruebas de aceptación y pruebas pilotos. También deben realizarse en este periodo la capacitación y acompañamiento a clientes para asegurar que adquieran los conocimientos necesarios en la manipulación del software.

El despliegue del software es una secuencia controlada y coordinada de acciones donde la solución pasa de la organización que ejecuta el desarrollo inicial a la organización encargada del mantenimiento.

Elementos a tener en cuenta:

- ✓ Riesgos asociados con la instalación de la nueva versión del sistema.

- ✓ Disponer de un plan de despliegue detallado.

2.3.9 Soporte

Durante esta etapa y por un tiempo limitado el proyecto ofrecerá un servicio para resolver conflictos y problemas de usabilidad y rendimiento del software entregado al cliente, suministrándole actualizaciones y parches a errores.

Elementos a tener en cuenta:

- ✓ Determinar qué hace el producto de software, su estructura y organización.
- ✓ Correcciones de prioridades bajas, de propuestas de modificación o informes de problemas.
- ✓ Disponibilidad de herramientas y el acceso al código fuente.

2.4 Guía de actividades a realizar por etapas.

Etapas	Actividades
Estudio Preliminar	Impartir capacitaciones: ✓ Elaborar una capacitación para asegurar que todos los miembros del proyecto adquieran los conocimientos necesarios que se definen en esta guía para asegurar esta característica.
Modelación del Negocio	Reglas del negocio: ✓ Deben ser explícitas, expresadas en términos sencillos, construidas a partir de hechos, éstos se definen a partir de conceptos, los que a su vez se representan por medio de términos empleando lenguajes de modelado como Lenguaje Unificado de Modelado y Notación para el Modelado de Procesos de Negocio. ✓ Utilizar herramientas para la gestión de las reglas de negocio como Oracle Motor de Reglas (del inglés Oracle Rules Engine), Microsoft

	Motor de Reglas del Negocio (del inglés Microsoft Business Rules Engine), para configurar las necesidades del negocio a través de la definición de objetos y reglas del negocio, definiendo responsabilidades a los distintos cargos de la organización repartiendo así el trabajo equitativamente y cuantitativamente, cuando, quien y donde tiene que desempeñar la tarea asignada.
Requisitos	Especificación de Requisitos de Software: ✓ Los requisitos descritos deben ser completos, consistentes y no ambiguos, precisos, verificables, modificables y que puedan ser leídos. ✓ Deben estar descritos de tal manera que no describa aspectos del área de diseño o de implementación. ✓ Utilizar técnicas para realizar la captura de requisitos como la Introspección en la cual se recomienda que se ponga en lugar del cliente y trate de imaginar cómo desearía el cliente el sistema o el uso de entrevistas a grupos de desarrollo con el cliente con el objetivo de discutir sobre su problemática y llegar en conjunto a la Especificación de Requisitos. ✓ Identificación y definición de funciones del programa, sobre todo las opcionales. ✓ Granularidad de los requerimientos ya que esto afecta a la dificultad o trazabilidad, se puede traducir por: obtener el mayor grado de detalle en el análisis de requerimientos.
	Elección de patrones de diseño:

Análisis y Diseño	✓ La correcta elección de la estructura del programa, su división en entidades y el flujo de datos entre ellos. Como en otras actividades, es importante usar los conocimientos sobre procesamiento de datos que tenga el equipo ya que esto puede revelar posibilidades importantes sobre la reutilización de partes de programas existentes o bibliotecas de funciones que ya han demostrado su utilidad. ✓ Examinar las debilidades del lenguaje de programación a utilizar para evitar problemas durante el desarrollo. ✓ Utilizar patrones para facilitar la reutilización como el patrón Creador y Controlador. ✓ El diseño modular, combinado con el análisis arriba-abajo (del inglés top-down), y una adecuada documentación, permitiendo añadir y quitar cosas fácilmente, son las dos principales características que harán que se continúe cumpliendo los requerimientos de mantenibilidad. Esta actividad produce una descripción exacta y detallada de las funciones necesarias para completar la solución de programación propuesta. Exactitud y organización lógica de los datos: ✓ Poner las bases de datos en forma normal, es decir la especificación de las interfaces de usuario. Elección de estilos arquitectónicos: ✓ Definir patrón de organización general. ✓ Definir tipos de componentes presentes en el estilo.

	✓ Identificar interacciones que se establecen entre los componentes.
Implementación	Garantizar la legibilidad del código: ✓ En un programa debe haber nombres significativos tanto para variables, constantes, tipos, funciones y para facilitar su entendimiento debe haber código bien comentado. Disponer de la documentación generada y artefactos relacionados con la implementación: ✓ Documentación de diseño, requisitos aprobados y controlados. ✓ Una planificación de desarrollo detallada, incluyendo todas las revisiones de código que se harán y a qué nivel. Uso de las buenas prácticas de programación: ✓ Evitar el código no estructurado, modularizar el programa al máximo. ✓ Métricas de diseño que podrían ser aplicables durante la implementación como las de Bansiya y Davis, que permiten medir la cohesión de las clases, el encapsulamiento de los atributos protegidos, la complejidad entre los métodos, entre otras. ✓ Lograr una validación exhaustiva de los datos de entrada como el tamaño de los números, en las cadenas el tamaño mínimo y máximo, así como la presencia de caracteres especiales, el empleo de expresiones regulares, entre otros pueden disminuir considerablemente futuros problemas. ✓ Adición de comentarios que para esclarecer las funcionalidades

	desarrolladas mediante el uso de estándares de codificación establecidos por el equipo de arquitectura del proyecto. Detección de errores en el código: ✓ Detección de errores en el diseño detallado, si no se hace eso se puede llegar a perder tiempo. ✓ Uso de técnicas que faciliten el seguimiento de errores, como tracear el programa.
Prueba	Disponer de la documentación y artefactos: ✓ Manuales de usuario. ✓ Plan de pruebas detallado. ✓ Plantillas de documentación de las pruebas. ✓ Procedimientos para resolver anomalías. ✓ Procedimientos para la gestión de la configuración del software. ✓ Criterio para los resultados de las pruebas de sistema. ✓ Resultados de las pruebas a bajo nivel. ✓ Todas las pruebas de integración y de unidad deben estar documentadas. ✓ Se debe comprobar que la documentación técnica y de usuario han sido actualizadas. ✓ Los casos de prueba usados durante el desarrollo de software deberían guardarse para hacer un análisis de regresión después

	de las modificaciones. Métodos de evaluación: ✓ Realización de pruebas de regresión para evitar la introducción de errores durante las modificaciones. ✓ Seleccionar el método de evaluación correcto con el fin de identificar los estilos arquitectónicos o enfoques arquitectónicos utilizados. ✓ Utilizar métricas centradas en medir la característica de calidad mantenibilidad. (Ver Anexo 3)
Despliegue	Riesgos asociados con la instalación de la nueva versión del sistema: ✓ Asegurar el mínimo impacto sobre los usuarios del sistema debido a fallos del software imprevistos, no detectados durante las pruebas. ✓ Verificar correspondencia del escenario de despliegue con los requisitos no funcionales. ✓ Planificar y documentar los procedimientos de instalación alternativos. ✓ Contar con el acceso de copias de datos referentes al sistema sustituido. ✓ Organización de la capacitación en caso de que sea necesario. Esto abarca la relación de todos los recursos imprescindibles para dar cursos y preparaciones.

	Disponer de un Plan de Despliegue: ✓ Deberá incluir factores críticos en el tiempo, por ejemplo, las fechas disponibles para la instalación, así como los procedimientos de recuperación o vuelta atrás.
Soporte	Determinar qué hace el producto de software, su estructura y organización: ✓ Leer la documentación relacionada con el software, discutir sobre el producto con los desarrolladores y operar con el producto de software. ✓ Revisar las especificaciones, revisar la estructura general, analizar y producir árboles de llamadas, hacer presentaciones orales para el resto del personal de soporte y añadir comentarios al código. ✓ Actualizar documentos como especificaciones, manuales de soporte para programadores, manuales de usuarios o guías para la instalación. Corrección de prioridades bajas, de propuestas de modificación o informes de problemas: ✓ Identificar y corregir cuando alguien otorgue una prioridad baja a un problema y en realidad sea un problema de prioridad baja. Disponibilidad de herramientas y el acceso al código fuente: ✓ Disponibilidad de herramientas para análisis de código, capacidad para operar con el producto para determinar capacidades y disponibilidad de un entorno para las pruebas.

Tabla 2.2 Guía de actividades a realizar por etapas

Conclusiones parciales

En el presente capítulo se dio cumplimiento al objetivo de la investigación relacionado con la definición de acciones o actividades a realizar a lo largo del ciclo de desarrollo de software, para asegurar la Mantenibilidad en el producto final. Las actividades se describen considerando las acciones que se debían realizar en cada una de las etapas definidas. Se debe resaltar que aun cuando el procedimiento se enfocó en el aseguramiento de la mantenibilidad de software en los proyectos de CEGEL, este constituye una guía que puede ser mejorada considerando las peculiaridades de cada nuevo entorno.

CAPÍTULO 3: VALIDACIÓN DE LA PROPUESTA

3.1 Introducción

Este capítulo está dirigido a la evaluación de la propuesta de solución a través del juicio crítico expresado por cada especialista seleccionado para comprobar la validez y adecuación de la propuesta a través del instrumento utilizado, para evaluar el cumplimiento del objetivo general planteado en la presente investigación.

El criterio de especialistas es un método rápido y eficaz por el potencial que posee para conformar, valorar y enriquecer criterios, guías, concepciones, modelos, estrategias, metodologías y es por esto que se utiliza para la evaluación de este trabajo. Tiene la ventaja de adquirir información de interés con un cuestionario previamente elaborado, y su utilización permitió conocer la opinión y valoración de los especialistas seleccionados en una muestra sobre el asunto dado, para posteriormente conocer el grado de concordancia entre los especialistas, con respecto a las evaluaciones que hicieron mediante el coeficiente de Kendall.

3.2 Procedimiento utilizado para la aplicación del criterio de especialistas.

1. Primeramente se debe determinar el área del conocimiento que los especialistas deben dominar y así como determinar la población de especialistas que es apta para evaluar la propuesta y escoger una muestra a la cual se le aplicará el cuestionario anteriormente mencionado.

- En este caso el área del conocimiento seleccionada es Soporte de Software ya que es donde más conocimiento poseen sobre la característica que se trata en esta investigación.
- Población seleccionada: especialistas en soporte de software de la Universidad de las Ciencias Informáticas.
- Muestra: 5 especialistas en soporte de software de la Universidad de las Ciencias Informáticas.

2. Luego se identifican a través de un muestreo intencional los 5 especialistas que evaluarán la propuesta, de acuerdo al nivel de competencia en esta rama, así como su disposición y disponibilidad para contribuir con la investigación realizada.

- Nivel de competencia de los especialistas.

Para la selección de los especialistas que harán la valoración de la propuesta se escogieron un conjunto de indicadores generales que permitieron obtener información más rica y actualizada, estos fueron: la labor que desempeñan actualmente, los años de experiencia en el tema de Soporte de Software y los conocimientos teóricos que tienen del tema.

Los especialistas han trabajado directamente para realizar modificaciones en los productos desarrollados, así como en despliegues de estos para satisfacer las necesidades de los clientes tanto nacionales como internacionales. Uno de ellos lleva ocho años de experiencia como especialista en soporte de software, otro posee tres años, dos de ellos con dos años y el restante con un año de experiencia.

De acuerdo al cuestionario aplicado se obtuvo el siguiente resultado con respecto a la autovaloración del nivel de competencia de cada uno de los especialistas:

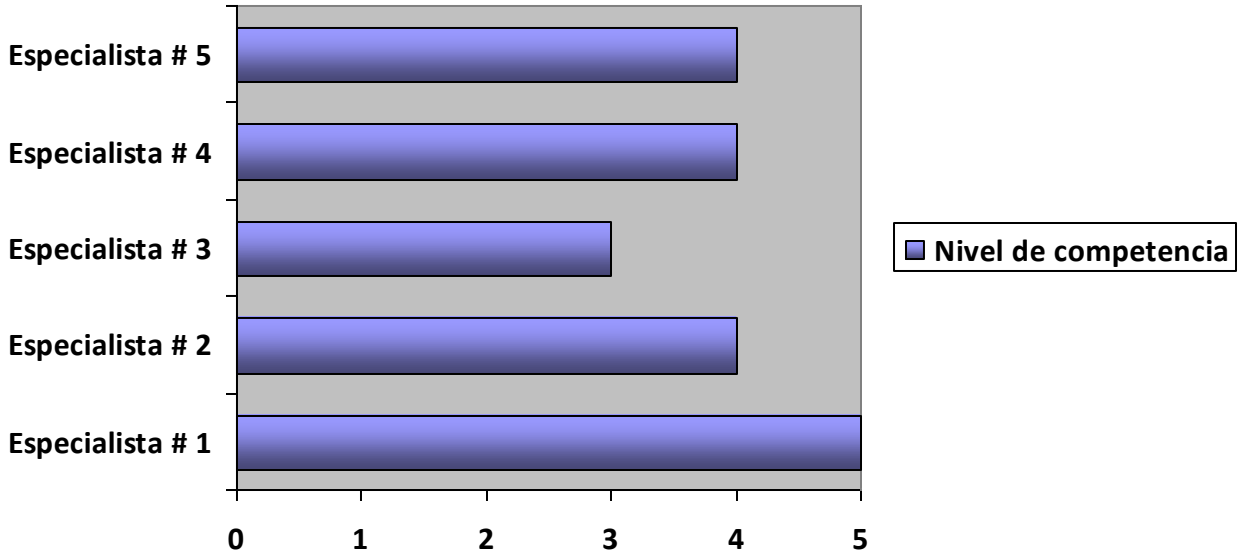

Figura 3.1 Nivel de autovaloración de los especialistas en una escala del 1 al 5.

En la misma se evidencia que en una escala del 1 al 5, el promedio del nivel de competencia que poseen los especialistas es de 4. Estos datos demuestran un alto nivel de competencia de los mismos, lo que da un alto valor a sus criterios con respecto a cada una de las preguntas realizadas en el cuestionario.

Para conocer 2 aspectos fundamentales como son los conocimientos teóricos del tema y la experiencia obtenida en la actividad práctica, el instrumento aplicado proporcionó la siguiente información (Figura 3.2):

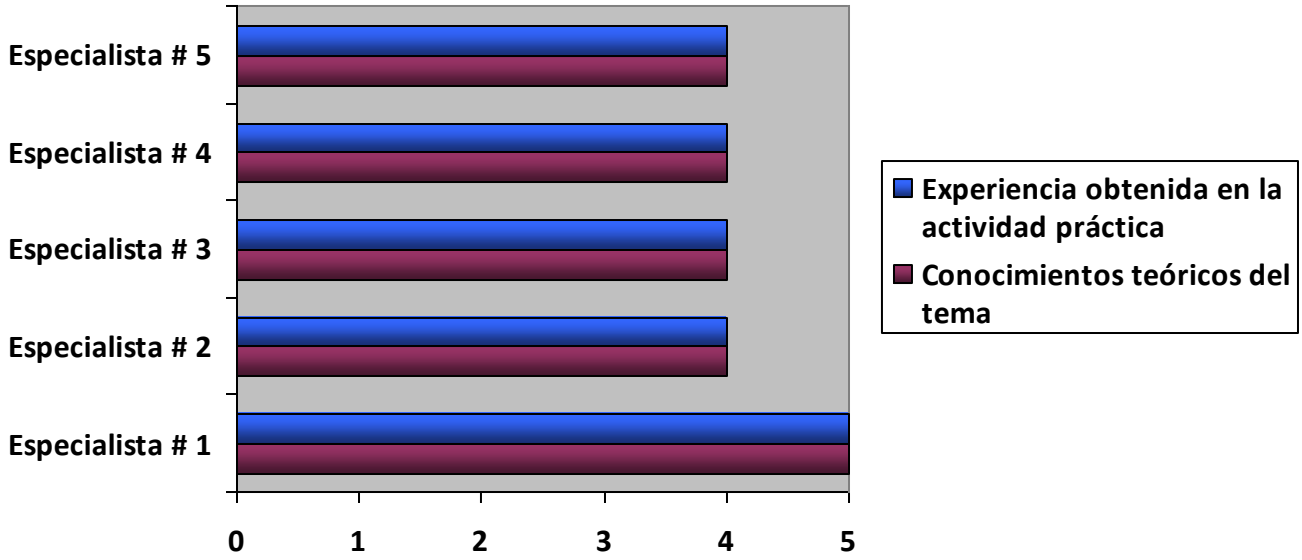

Figura. 3.2 Nivel de competencia de los especialistas.

En la misma se evidencia que en una escala del 1 al 5, el promedio de conocimiento teórico que poseen los especialistas es de 4,2. En cuanto a la experiencia en la actividad práctica el promedio es de 4,2. Estos datos demuestran un alto nivel de competencia de los mismos, lo que da alto valor a sus criterios con respecto a cada una de las preguntas realizadas en el cuestionario.

3. Luego se analizan los resultados de la aplicación del cuestionario donde se evalúa por parte de los especialistas cada elemento definido en la solución.

-En el cuestionario se formulan ocho preguntas dirigidas a valorar los elementos definidos para asegurar la mantenibilidad, durante las etapas del ciclo de desarrollo de software basado en el programa de mejora que lleva a cabo la universidad.

Las respuestas a las preguntas han sido graficadas de manera que se puedan comprender mejor los resultados en la tabla 3.1.

	Muy adecuado	Bastante adecuado	Adecuado	Poco adecuado	No adecuado
Pregunta # 1	4	1			
Pregunta # 2	3	1	1		
Pregunta # 3	2	2	1		
Pregunta # 4	3		2		
Pregunta # 5	4	1			
Pregunta # 6	2	2	1		
Pregunta # 7	1	2	2		
Pregunta # 8	3	2			

Tabla 3.1 Evaluación por preguntas de los especialistas.

Es válido aclarar que el proceso de evaluación es iterativo, o sea, se realizan varias rondas hasta que haya un consenso entre los especialistas, en este caso solo fue necesario realizar una ronda, ya que los criterios emitidos por los especialistas se encontraban desde Muy adecuado hasta Adecuado, no hubo criterios negativos. Hay que decir además, que el cuestionario se le envía a cada uno por separado para no condicionar las respuestas de los demás. Los datos que se muestran a continuación están referidos a la ronda que se realizó. A continuación se muestra toda la información recopilada a partir de la aplicación del instrumento.

3.1 Resultados de la aplicación del cuestionario

En respuesta a la primera pregunta del cuestionario, Impartir capacitaciones relacionadas con el aseguramiento de la mantenibilidad, como elemento fundamental en la etapa de Estudio Preliminar que influye en la mantenibilidad del sistema, los especialistas evaluaron lo siguiente:

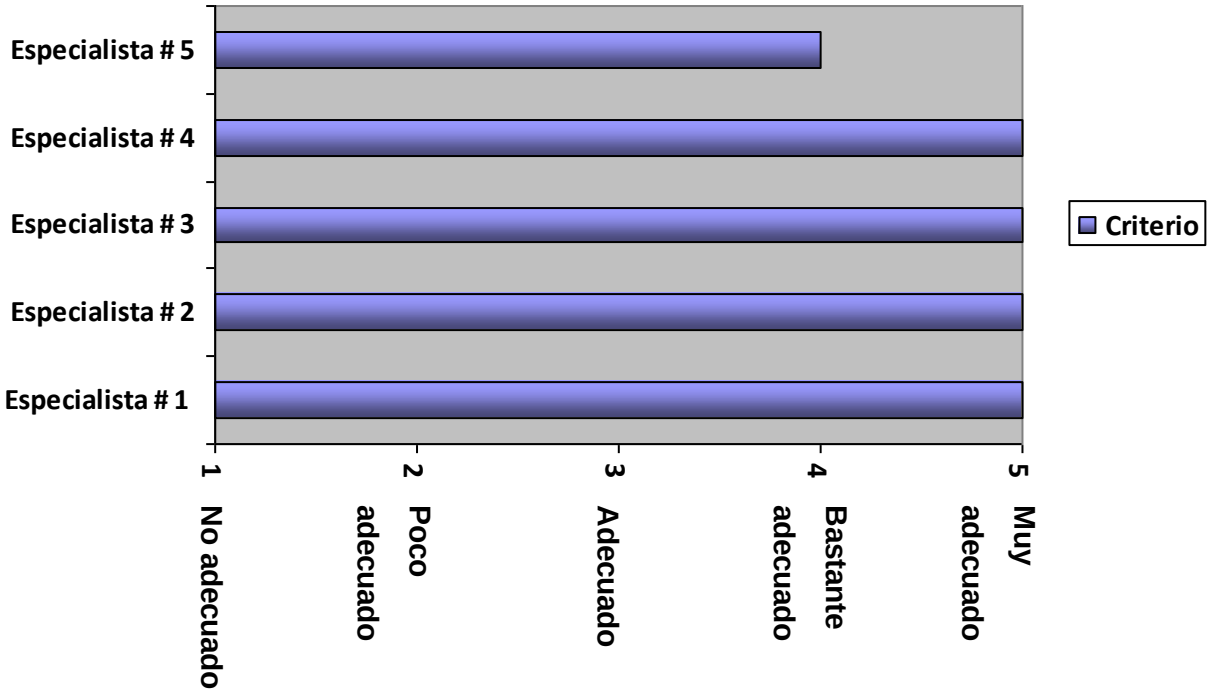

Fig. 3.3 Criterio de evaluación de la pregunta 1 por especialista.

El 100 % de los especialistas valora entre Muy adecuado y Bastante adecuado los elementos definidos en la etapa de Estudio Preliminar como se aprecia en la fig. 3.4.

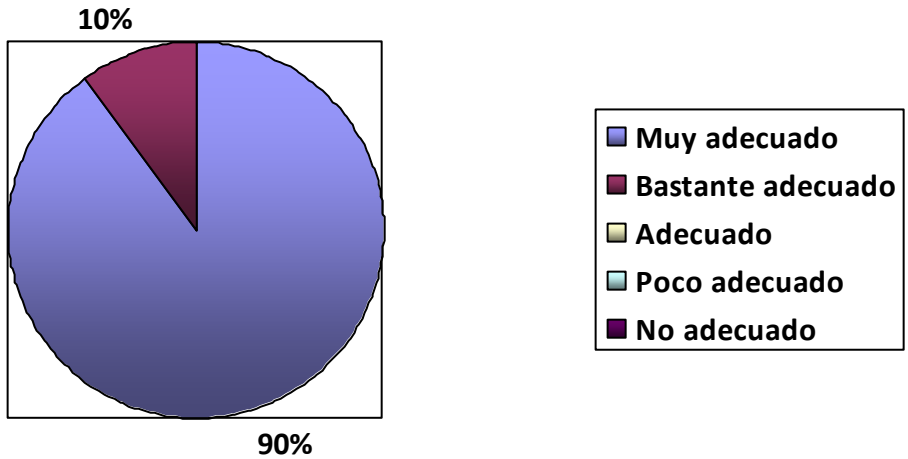

Fig. 3.4 Porciento del criterio de evaluación de los especialistas para la pregunta 1.

En respuesta a la segunda pregunta, referente a la definición de las Reglas del negocio durante la etapa de Modelación del Negocio, los especialistas evaluaron lo siguiente:

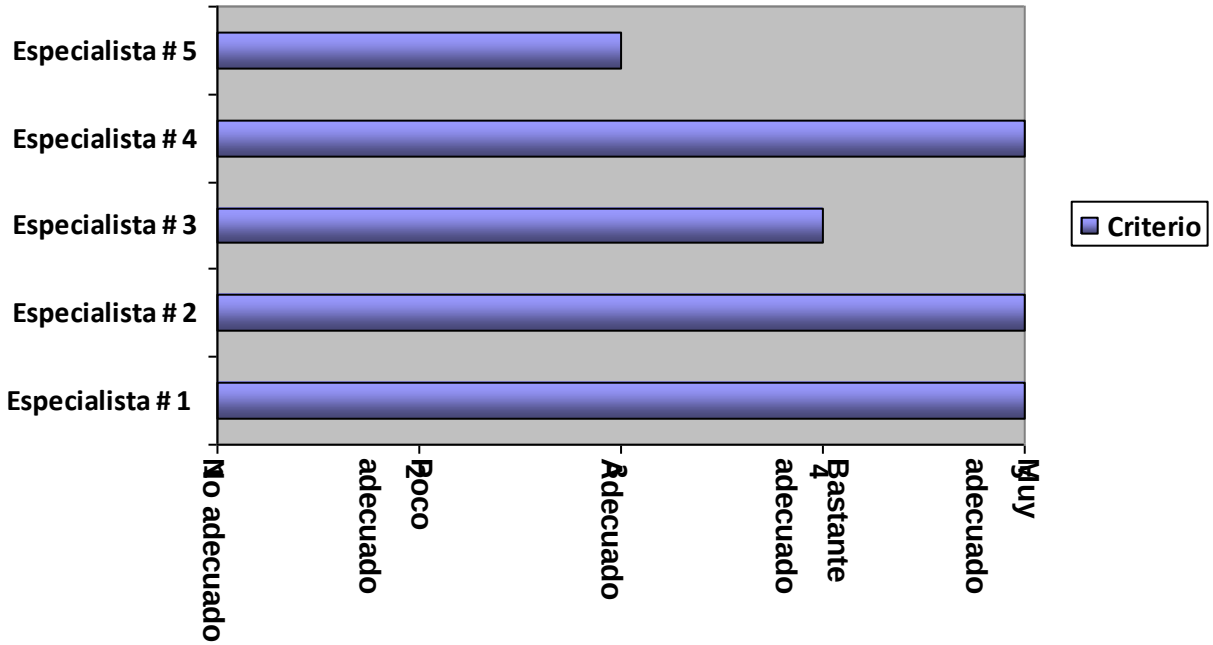

Fig. 3.5 Criterio de evaluación de la pregunta 2 por especialista

El 100 % de los especialistas valora entre Muy Adecuado y Adecuado los elementos definidos en la etapa de Modelación del Negocio como se aprecia en la fig. 3.6.

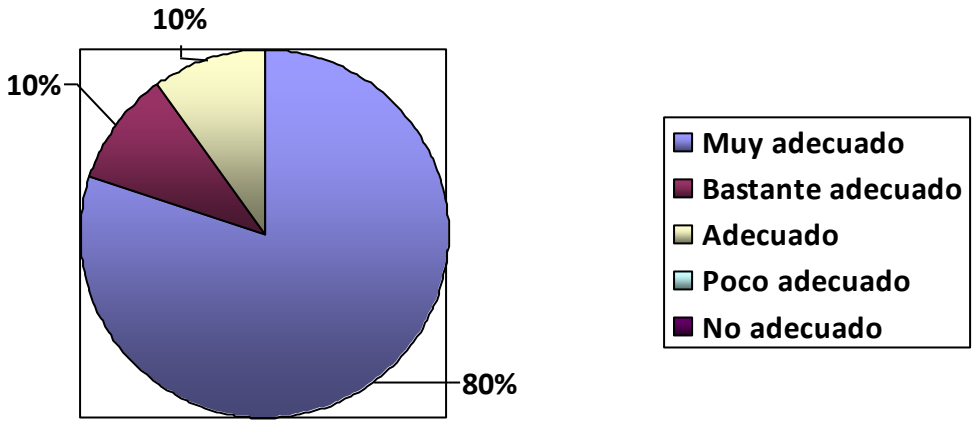

Fig. 3.6 Porciento del criterio de evaluación de los especialistas para la pregunta 2.

La tercera pregunta, referente a la Especificación de Requisitos de Software como elemento fundamental en el aseguramiento de la mantenibilidad durante la etapa de Requisitos. Los especialistas evaluaron su criterio respecto a esta interrogante:

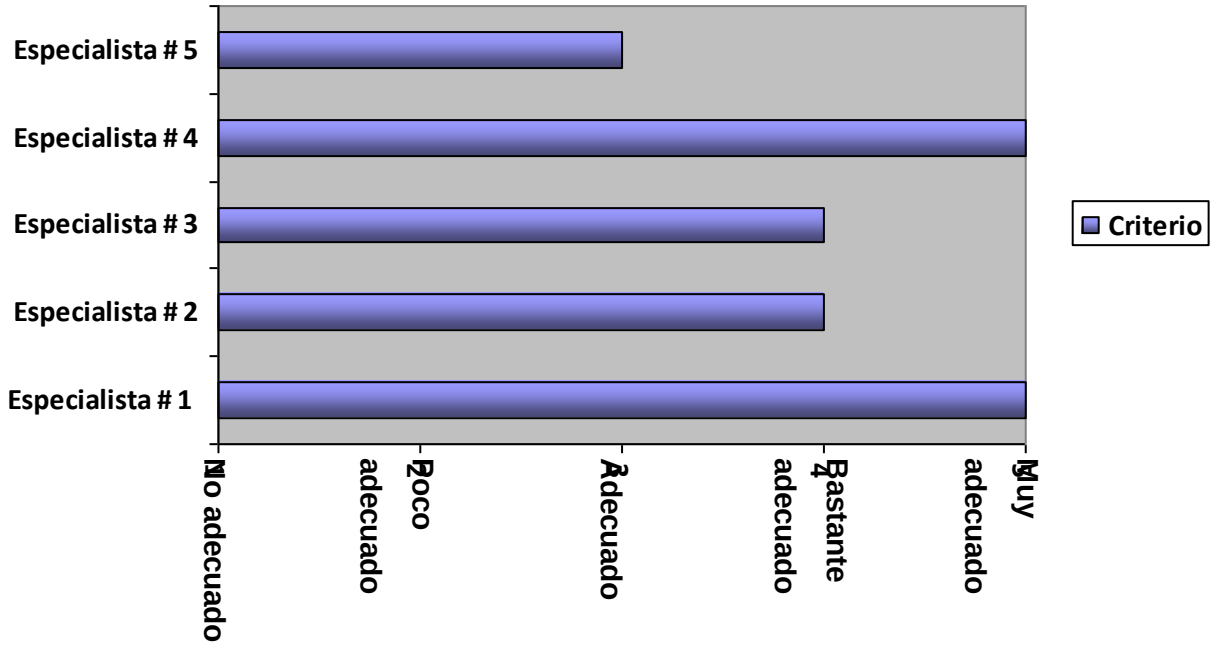

Fig. 3.7 Criterio de evaluación de la pregunta 3 por especialista.

El 100 % de los especialistas valora entre Muy Adecuado y Adecuado los elementos primordiales de la etapa de Requisitos como se aprecia en la fig. 3.8

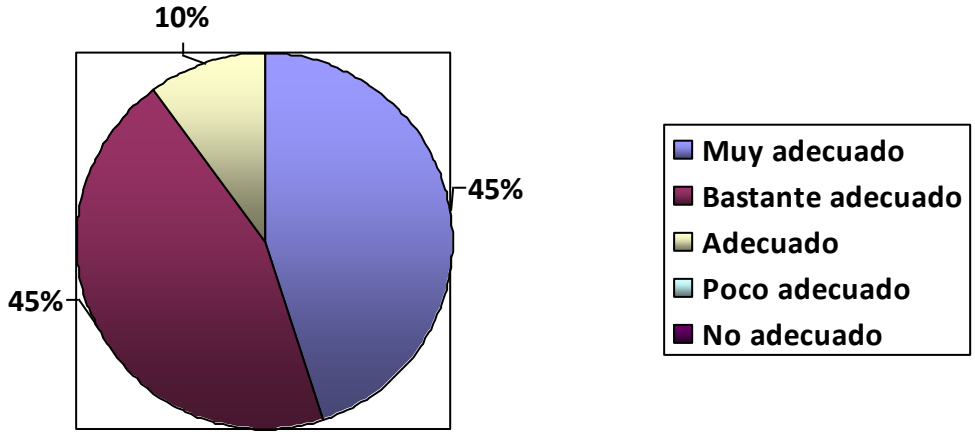

Fig. 3.8 Porciento del criterio de evaluación de los especialistas para la pregunta 3.

En respuesta a la cuarta pregunta, referente a la Elección de patrones de diseño, Elección de estilos arquitectónicos, Exactitud y organización lógica de los datos como elementos primordiales en el aseguramiento de la mantenibilidad durante la etapa de Análisis y Diseño, los especialistas alegan lo siguiente:

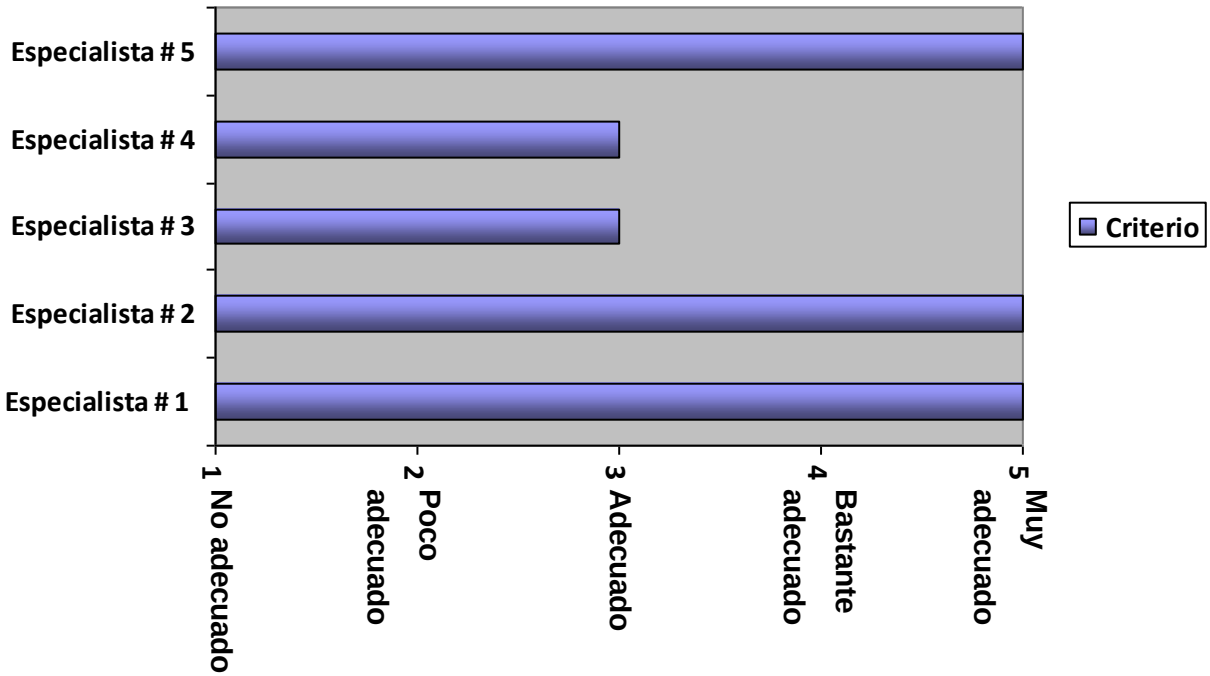

Fig. 3.9 Criterio de evaluación de la pregunta 4 por especialista.

El 100 % de los especialistas valora entre Muy adecuado y Adecuado los elementos definidos en la etapa de Análisis y Diseño como se aprecia en la fig. 3.10.

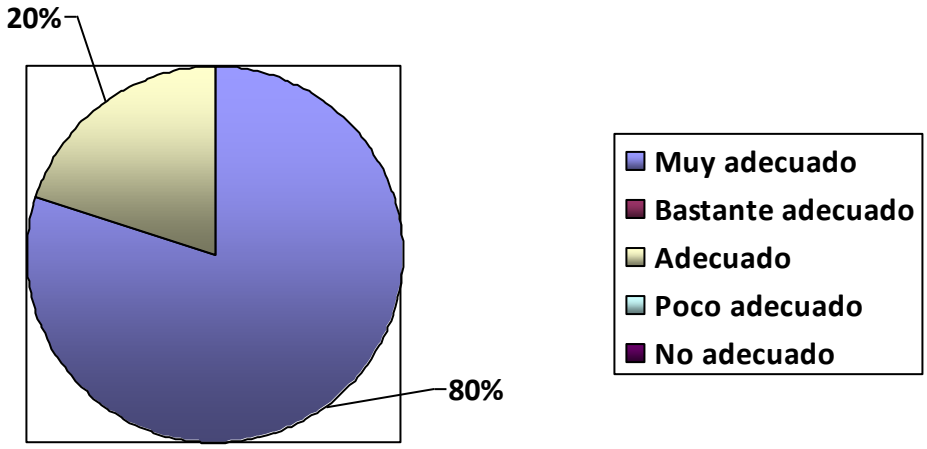

Fig. 3.10 Porciento del criterio de evaluación de los especialistas para la pregunta 4.

La quinta pregunta, referente a Disponer de la documentación generada y artefactos relacionados con la implementación, Garantizar la legibilidad del código, Uso de las buenas prácticas de programación y Detección de errores en el código como elementos críticos que influyen en el aseguramiento de la mantenibilidad. En esta pregunta los especialistas evaluaron lo siguiente:

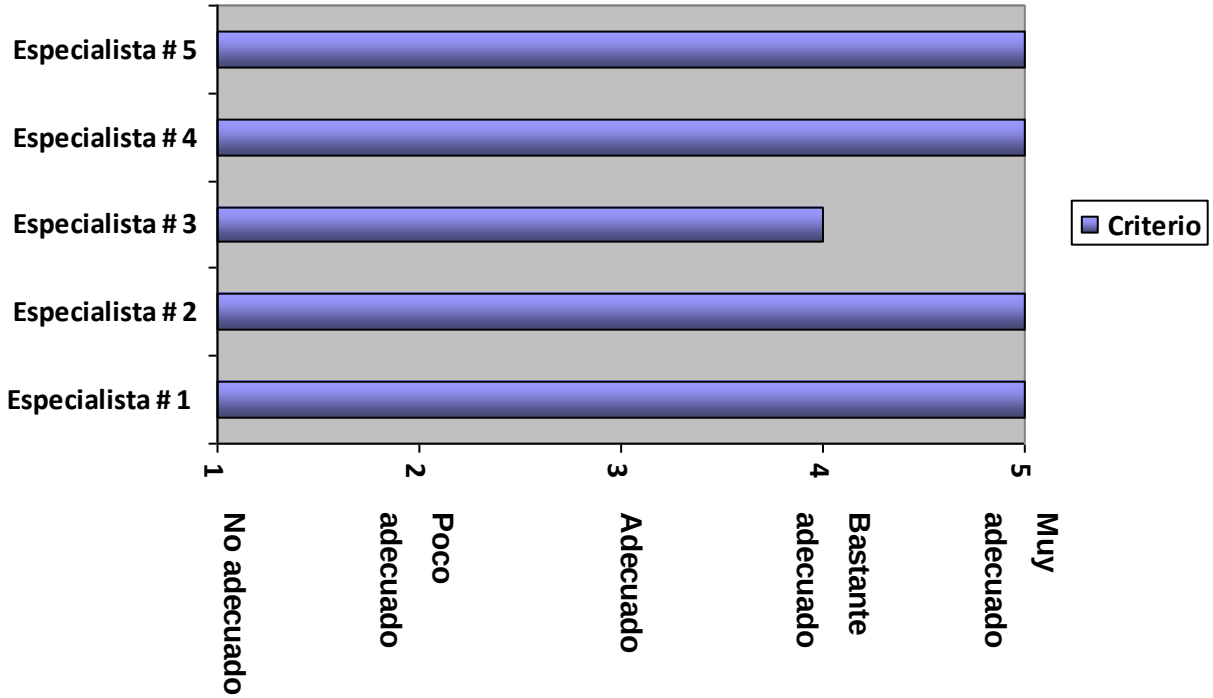

Fig. 3.11 Criterio de evaluación de la pregunta 5 por especialista.

El 100 % de los especialistas valora entre Muy Adecuado y Bastante adecuado los elementos de la etapa de Implementación como se aprecia en la fig. 3.12.

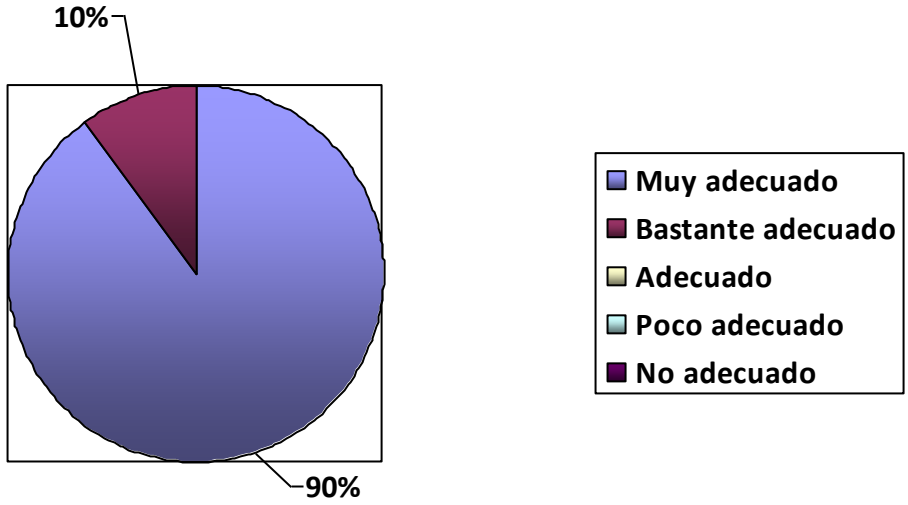

Fig. 3.12 Porciento del criterio de evaluación de los especialistas para la pregunta 5.

En la sexta pregunta, referente a Disponer de toda la documentación generada y artefactos, además de la Selección de métodos de evaluación arquitectónicos durante la etapa de Prueba como elementos fundamentales para asegurar la mantenibilidad, los especialistas respondieron lo siguiente:

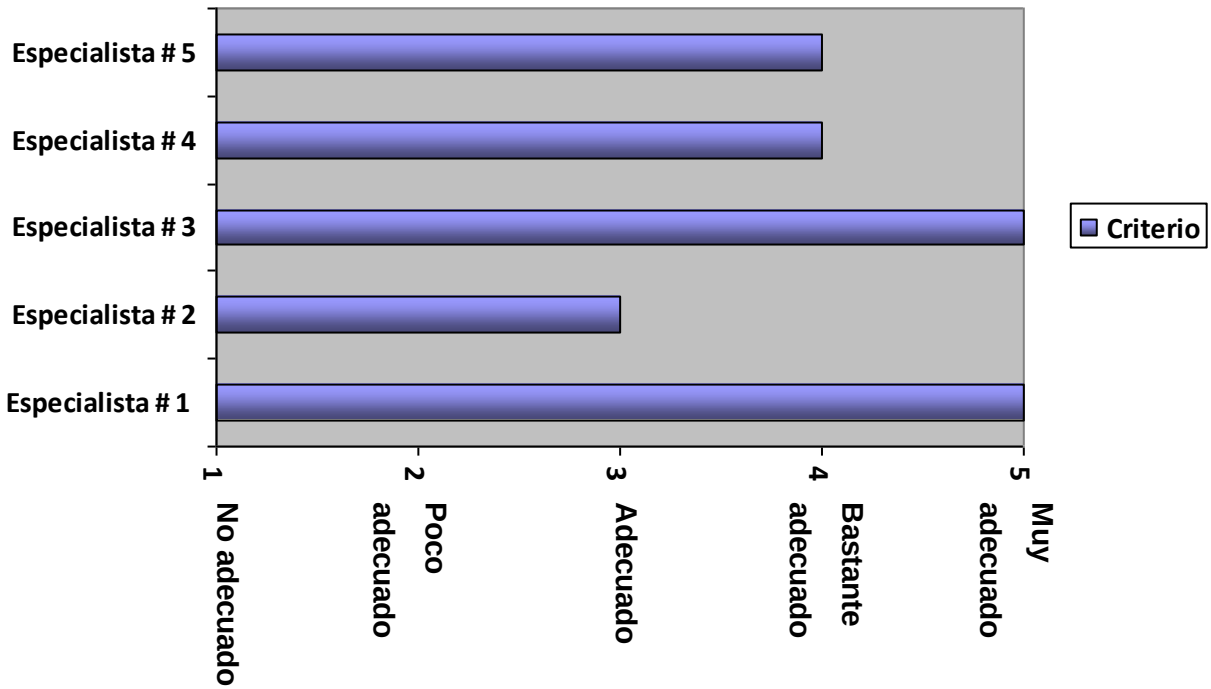

Fig. 3.13 Criterio de evaluación de la pregunta 6 por especialista.

El 100 % de los especialistas valora entre Muy Adecuado y Adecuado los elementos de la etapa de Prueba como se aprecia en la fig. 3.14

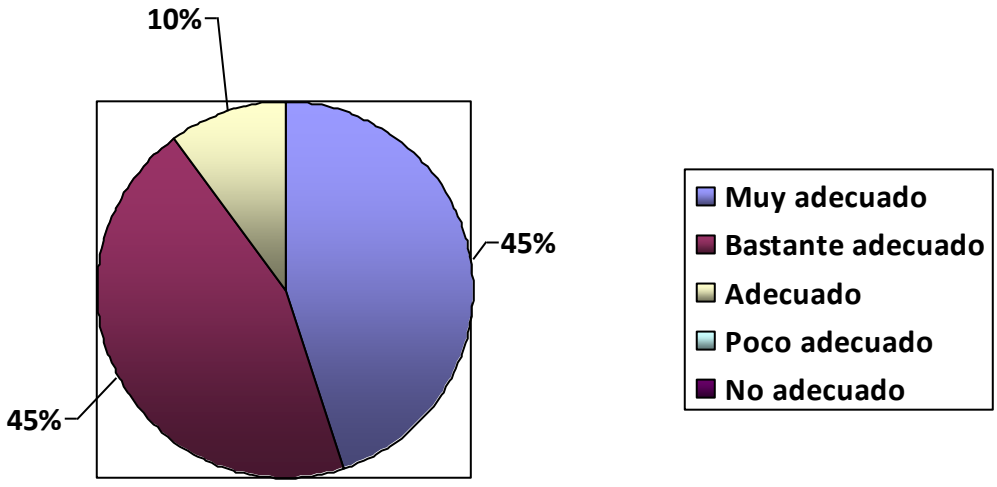

Fig. 3.14 Porciento del criterio de evaluación de los especialistas para la pregunta 6.

En la séptima pregunta, referente a Riesgos asociados con la instalación de la nueva versión del sistema, y Disponer de un plan de despliegue detallado como elementos fundamentales durante la etapa de Despliegue que influyen en la mantenibilidad del sistema, los especialistas evaluaron lo siguiente:

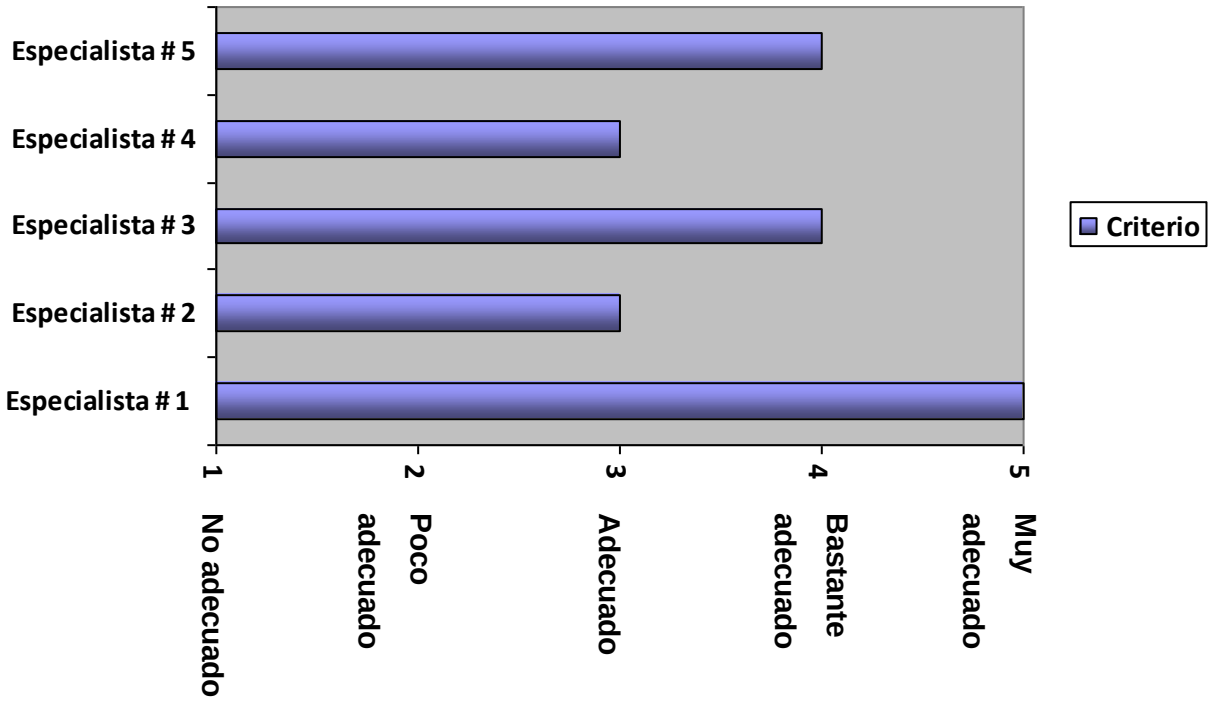

Fig. 3.15 Criterio de evaluación de la pregunta 7 por especialista.

El 100 % de los especialistas valora entre Muy Adecuado y Adecuado los elementos de la etapa de Despliegue como se aprecia en la fig. 3.16

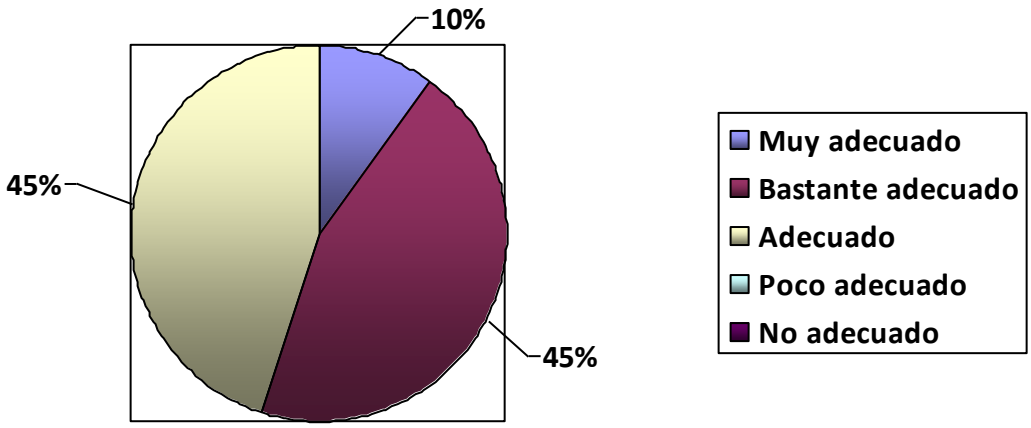

Fig. 3.16 Porciento del criterio de evaluación de los especialistas para la pregunta 7.

En respuesta a la octava y última pregunta de la encuesta referente a Determinar qué hace el producto de software, su estructura y organización, Correcciones de prioridades bajas, de propuestas de modificación o informes de problemas y Disponibilidad de herramientas y el acceso al código fuente como elementos fundamentales que influyen en la mantenibilidad durante la etapa de Soporte, los especialistas evaluaron lo siguiente:

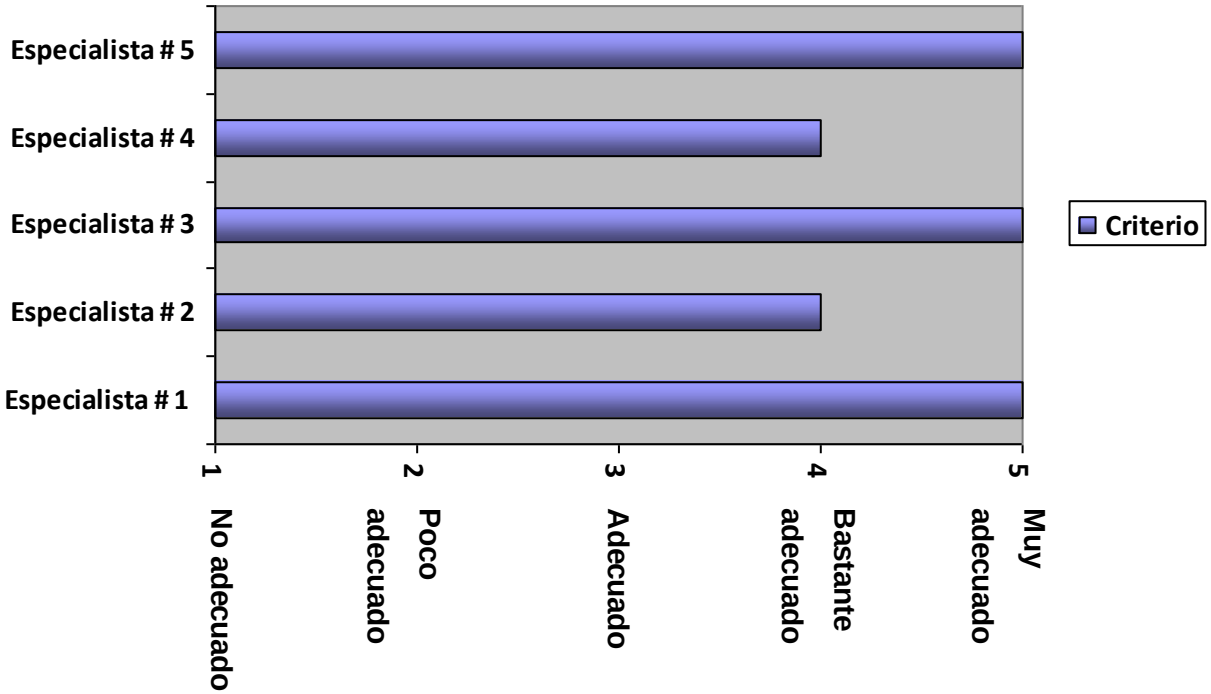

Fig. 3.17 Criterio de evaluación de la pregunta 8 por especialista.

El 100 % de los especialistas valora entre Muy Adecuado y Bastante adecuado los elementos de la etapa de Soporte como se aprecia en la fig. 3.18.

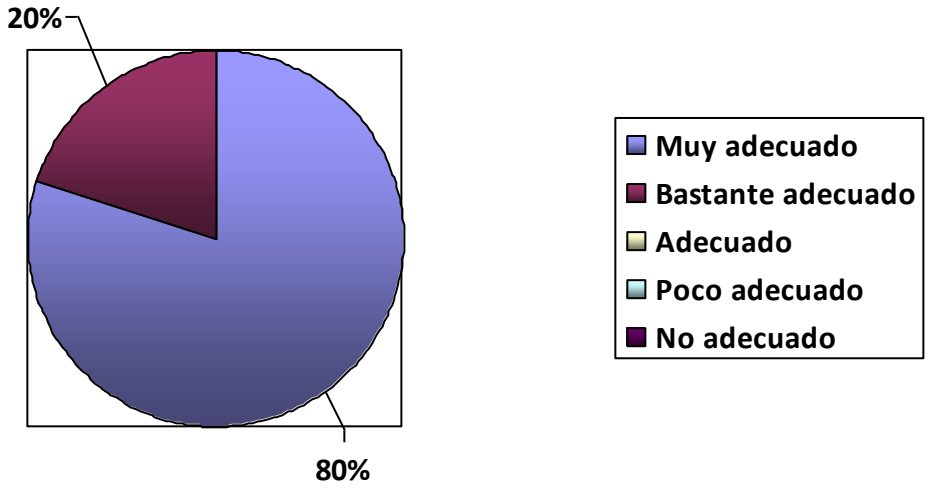

Fig. 3.18 Porciento del criterio de evaluación de los especialistas para la pregunta 8.

4. Procesamiento estadístico de los datos obtenidos.

4.1 Establecimiento de la concordancia entre los especialistas mediante el coeficiente de Kendal.

Para realizar el grado de coincidencia se construye una tabla de aspectos a evaluar contra especialistas donde se sitúan los rangos de valoración de cada aspecto evaluado por cada uno de los especialistas; estos datos son tomados de los resultados de la encuesta de validación, que está resumida en la tabla 3.2.

Especialistas	Preguntas							
	P1	P2	P3	P4	P5	P6	P7	P8
E1	5	5	5	5	5	5	5	5
E2	5	5	4	5	5	3	3	4
E3	5	4	4	3	4	5	4	5
E4	5	5	5	3	5	4	3	4
E5	4	3	3	5	5	4	4	5

Tabla 3.2 Resultados obtenidos al aplicar la encuesta a los especialistas.

Como se puede evidenciar por lo antes mostrado los resultados arrojados por la encuesta demuestran la aceptación de los elementos críticos por parte de los especialistas, pero para mayor confirmación se analizan los resultados estadísticamente, para determinar el grado de concordancia entre los especialistas mediante el coeficiente de Kendall (W) y la prueba de hipótesis. Para ello se utilizó herramienta Paquete Estadístico para las Ciencias Sociales (SPSS) para la explotación de los datos, obteniéndose como resultado:

Kendall's W Test

Test Statistics

N	5
Kendall's W[a]	,806
Chi-Square	7,194
df	7
Asymp. Sig.	,019

a. Kendall's Coefficient of Concordance

Tabla 3.3 Resultados estadísticos.

Esta tabla muestra el número de casos válidos (N), el valor del estadístico W (W de Kendall), su transformación en Chi-cuadrado (Chi-Square), sus grados de libertad (df) y el nivel crítico (Asymp. Sig.).
Para saber si W de Kendall es significativamente distinta de 0 se realizó una prueba de hipótesis donde:
H0 = No hay concordancia entre los especialistas.
H1 = Hay concordancia entre los especialistas.
El coeficiente W de Kendall es una medida de la concordancia de los especialistas y por definición del Método Delphi, el resultado debe moverse en un rango de 0 a 1 y debe ser siempre W > 0,5 porque cuanto más se acerque el coeficiente a 1, mayor será el grado de concordancia entre los especialistas. En el estudio realizado resultó ser un aproximado de W= 0,80, por tanto la propuesta resultó ser aceptada y con un nivel de concordancia alto con respecto a los criterios que fueron evaluados.
La tabla muestra que el valor del nivel crítico (0,019) es menor que 0,05, por tanto se puede rechazar la hipótesis de concordancia nula y concluir que entre las puntuaciones de las variables estudiadas existe asociación significativa.

Conclusiones parciales

Con la validación de este instrumento se puede apreciar que arrojó un resultado positivo, ya que el 100 por ciento de los criterios sobre los elementos definidos en el ciclo de desarrollo se encontraban desde Adecuado hasta Muy adecuado y con un nivel de concordancia alto entre los especialistas, demostrando así que estos elementos pueden influir en el aseguramiento de la mantenibilidad en los sistemas.

CONCLUSIONES

Con la realización de este trabajo de diploma se arribó a las siguientes conclusiones:

- ✓ Se determinó a través de una entrevista que los integrantes de proyectos del Centro no utilizan ningún modelo para asegurar la mantenibilidad en sus productos.

- ✓ Se seleccionó satisfactoriamente el modelo de calidad que sirvió como guía para el desarrollo de la propuesta de solución.

- ✓ Se definieron por etapas del ciclo de vida actividades a tener en cuenta durante el desarrollo de los sistemas informáticos.

- ✓ Se pudo corroborar que los elementos definidos en la guía propuesta pueden influir en la mantenibilidad de los sistemas. La misma fue validada satisfactoriamente, con un 100 por ciento de los criterios entre Adecuado y Muy Adecuado.

- ✓ Se elaboró una guía para el aseguramiento de la mantenibilidad en los sistemas de gobierno electrónico, proponiéndose en correspondencia a las características de los proyectos del Centro.

RECOMENDACIONES

Tomando como punto de partida los resultados obtenidos con la realización de este trabajo de diploma, se recomienda:

- ✓ Aplicar la guía propuesta.
- ✓ Profundizar en el estudio y la investigación de los diferentes aspectos que influyen en la mantenibilidad en los sistemas informáticos.
- ✓ Hacer un plan de capacitación para todas las personas inmersas en el proceso.
- ✓ Realizar la evaluación de la mantenibilidad en los productos de software, haciendo uso de las diferentes métricas que se presentan en el modelo SQuaRE, para obtener el nivel de calidad presente en esta característica.
- ✓ Actualizar la guía de forma periódica, de acuerdo a lo que se defina en cada centro.

BIBLIOGRAFÍA

Calisoft. 2011. *Programa de Mejora.* Universidad de las Ciencias Informáticas. 2011.

Cueva Lovelle , Juan Manuel. 1999. *Calidad del Software.* 1999. http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

Fuertes Castro, José Luis . 2008. *Calidad del Software.* 2008. Vol. IIMV, http://www.iimv.org/actividades2/08_Tecnologia/Jose%20LuisFuertesCalidad%20del%20Software.pdf.

Giraldo Rendón, Juan Pablo . Monografias.com S.A. [En línea] http://www.monografias.com/trabajos15/ingenieria-software/ingenieria-software.shtml.

González, Jorge Luis Valdés. 2011. *Metodología de la Investigación. Un enfoque orientado al pregrado.* CEGEL, Facultad 3, Universidad de las Ciencias Informáticas. 2011.

G. Piattini, Mario, Calero, Coral y Ángeles Moraga, Ma . 2010. *Calidad del Producto y Proceso del Software.* s.l. : Ra-Ma, 2010. ISBN: 9788478979615.

ISO/IEC. 2011. ISO/IEC 25010. *Systems and software engineering — Systems and software Quality, Requirements and Evaluation (SQuaRE) — System and software quality models.* Primera. 2011.

—. 2002. Software engineering –Product quality – Part 3: Internal. 2002.

Minguet Melián, Jesús Ma y Hernández Ballesteros, Juan Francisco. 2003. *La Calidad del Software y su Medida.* s.l. : CENTRO DE ESTUDIOS RAMON ARECES, S.A., 2003. ISBN: 8480046112.

M., Piattini. 2000. Mantenimiento del Software: Modelos, técnicas y métodos para la gestión del cambio. [ed.] Francisco Ruiz y Macario Polo. s.l. : Ra-Ma, 2000.

Qualitrain. 2012. Qualitrain. [En línea] 2012. http://www.qualitrain.com.mx/Aseguramiento-de-la-Calidad-de-Software.html.

Quispe-Otazu, Rodolfo. 2012. Computacion e Informatica. [En línea] 2012. http://www.rodolfoquispe.org/blog/que-es-la-calidad-de-software.php.

Rodríguez Monje, Moisés. 2011. *Calidad de Procesos y Productos de Software: Calidad del Producto Software.* Alarcos Quality Center. 2011.

Ruiz Morilla, José Joaquín . 2008. *ISO 9126 vs. SQuaRE.* Departamento de Tecnologías y Sistemas de Información, Escuela Superior de Informática de Ciudad Real. 2008.

Sicilia, Miguel Angel . 2012. Connexions. [En línea] 2012. http://cnx.org/content/m17457/latest/.

—. 2012. Connexions. [En línea] 2012. http://cnx.org/content/m17471/latest/.

—. 2012. Connexions. [En línea] 2012. http://cnx.org/content/m17461/latest/.

Sicilia, Miguel Angel. 2012. Connexions. [En línea] 2012. http://cnx.org/content/m17452/latest/?collection=col10583/latest.

Soto, Lauro. 2012. Tecnologico. [En línea] 2012. http://www.mitecnologico.com/Main/CalidadDelSoftware.

S. Pressman, Roger . 2010. *Ingenieria del Software: Un Enfoque Practico.* [ed.] Darrel Ince. Quinta. 2010.

Sridhar, SESHADRI. 2000. *Aseguramiento de la Calidad de Software.* 2000. http://www.qualitrain.com.mx/Aseguramiento-de-la-Calidad-de-Software.html.

Villagra, Sergio. 2006. [En línea] 2006. http://es.scribd.com/doc/69546265/WP03-Una-Introduccion-a-CMMI.

Anexo 1

Entrevista aplicada a los integrantes de proyectos del Centro CEGEL.

1. ¿Existe un responsable para llevar a cabo la mantenibilidad en el ciclo de vida del software del proyecto?____

2. ¿Cuáles son los principales estándares internacionales utilizados en la mantenibilidad del proyecto? ____

3. ¿Utilizan algunas métricas para medir la mantenibilidad? ____

4. ¿Utilizan alguna herramienta para ayudar al personal de mantenibilidad? ____

5. ¿Utilizan alguna metodología de desarrollo que les permita garantizar elementos que influyen en la mantenibilidad? ____

Anexo 2

Encuesta aplicada a especialista

Encuesta a especialistas para someter a sus criterios la propuesta de la Guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico.

Estimado profesor(a):

La presente encuesta forma parte de la aplicación del Método de Valoración de Especialistas. Con este fin solicitamos su valiosa colaboración, y de antemano le aseguramos, que sus opiniones se tendrán en cuenta para la aplicación de la Guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico.

Muchas Gracias.

Nombre y Apellidos: _____

Fecha de graduación: _____

Puesto de trabajo actual: _____

Calificación profesional:

Ingeniero___ Licenciado en Educación___Master___Doctor___

Años de experiencia como especialista en soporte de software: _____

Categoría Docente:

Prof. Instructor___ Prof. Asistente____ Prof. Auxiliar____

Prof. Titular____ Prof. Adjunto_____

1. Seleccione en una escala del 1 – 5 el valor que corresponda con el grado de conocimientos que usted posee acerca del tema de investigación que desarrollamos (aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico), considerando 1 como no tener ningún conocimiento y 5 el de pleno conocimiento de la problemática tratada.

1	2	3	4	5

2. Valore el grado de influencia que cada una de las fuentes que le presentamos a continuación ha tenido en su conocimiento y criterios sobre el tema que se investiga.

Fuentes de argumentación	**Grado de influencia de cada fuente**		
	ALTO	**MEDIO**	**BAJO**
Conocimientos teóricos que posee acerca del tema.			
Su experiencia obtenida en la actividad práctica.			

La propuesta de la guía para el aseguramiento de la mantenibilidad en los sistemas informáticos de Gobierno Electrónico se encuentra adjunta a esta encuesta. Para su análisis y mejor comprensión se le informa que consta con 2 partes fundamentales, la primera parte (columna izquierda) es la relación de las etapas del ciclo de vida de los proyectos y la segunda parte (columna derecha) son las actividades propuestas para cada etapa del ciclo de vida.

3. Le pedimos su criterio acerca del elemento definido en la etapa de Estudio Preliminar.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**
¿Considera usted que Impartir capacitaciones relacionadas con el aseguramiento de la mantenibilidad es un elemento fundamental en la etapa de Estudio Preliminar que influye en la mantenibilidad de los sistemas?					

4. Le pedimos su criterio acerca del elemento definido en la etapa de Modelación del Negocio.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**
¿Considera usted que la correcta definición de las Reglas del Negocio, es un elemento fundamental en la etapa de Modelación del Negocio que influye en la mantenibilidad de los sistemas?					

5. Le pedimos su criterio acerca de los elementos definidos en la etapa de Requisitos.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**
¿Considera usted que la correcta definición de la Especificación de Requisitos de Software es un elemento primordial en el aseguramiento de la mantenibilidad durante la etapa de Requisitos?					

6. Le pedimos su criterio acerca de los elementos definidos en la etapa de Análisis y Diseño.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**

¿Considera usted que la Elección de patrones de diseño y Estilos arquitectónicos, además de la Exactitud y organización lógica de los datos son elementos esenciales en la etapa de Análisis y Diseño que influyen en la mantenibilidad de los sistemas?					

7. Le pedimos su criterio acerca de los elementos definidos en la etapa de Implementación.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**
¿Considera usted que Disponer de la documentación generada y artefactos relacionados con la implementación, Garantizar la legibilidad del código, Uso de las buenas prácticas de programación y Detección de errores en el código, son elementos críticos para asegurar la mantenibilidad durante la etapa de Implementación?					

8. Le pedimos su criterio acerca de los elementos definidos en la etapa de Prueba.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**
¿Considera usted que Disponer de toda la documentación generada y artefactos, así como la Selección de métodos de evaluación arquitectónicos, son elementos fundamentales durante la etapa de Prueba para asegurar la mantenibilidad de los sistemas?					

9. Le pedimos su criterio acerca de los elementos definidos en la etapa de Despliegue.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**

¿Considera usted que los Riesgos asociados con la instalación de la nueva versión del sistema y Disponer de un plan de despliegue detallado, son elementos fundamentales que influyen en el aseguramiento de la mantenibilidad durante la etapa de Despliegue?					

10. Le pedimos su criterio acerca de los elementos definidos en la etapa de Soporte.

	CRITERIO DE ESPECIALISTA				
Pregunta	**C1 Muy adecuado**	**C2 Bastante adecuado**	**C3 Adecuado**	**C4 Poco adecuado**	**C5 No adecuado**

¿Considera usted correcto que Determinar qué hace el producto de software, su estructura y organización, Correcciones de prioridades bajas, de propuestas de modificación o informes de problemas, así como la Disponibilidad de herramientas y el acceso al código fuente, son elementos fundamentales durante esta etapa para asegurar la mantenibilidad en los sistemas?					

Anexo 3

Métricas de mantenibilidad

Métricas internas de mantenibilidad se utilizan para predecir el nivel de esfuerzo necesario para modificar el producto de software.

Métricas diagnósticabilidad

Las métricas internas de diagnósticabilidad indican un conjunto de atributos para predecir el esfuerzo o recursos requeridos por el personal de mantenimiento o usuarios al tratar de diagnosticar deficiencias o causas de fallas, para identificar partes que serán modificadas en el producto de software.

Métricas cambiabilidad

Las métricas internas de cambiabilidad indican un conjunto de atributos para predecir el esfuerzo requerido por el personal de mantenimiento o usuarios cuando se trata de implementar una modificación específica en el producto de software.

Métricas de estabilidad.

Las métricas internas de estabilidad indican un conjunto de atributos para predecir cuan estable es un producto de software ante cualquier modificación.

Métricas de capacidad de prueba.

Las métricas internas de capacidad de prueba indican un conjunto de atributos para predecir la cantidad de funciones de pruebas autónomas que han sido implementadas y que se encuentran presentes en el producto de software.

Métricas de mantenibilidad de cumplimiento.

Indicadores internos de cumplimiento relacionados con mantenimiento, indican un conjunto de atributos para evaluar la capacidad del producto de software para cumplir con los artículos tales como normas, convenciones y reglamentos de la organización de usuarios en relación con el mantenimiento de software.

Métrica de diagnósticabilidad	
Nombre de la métrica	Preparación de la función de diagnóstico
La métrica se propone medir	¿Cómo se completa la prestación de la función de diagnóstico?
Método de aplicación	Cuente el número de funciones de diagnóstico tal como se especifica y compara con el número de funciones de diagnóstico requeridas en las especificaciones.
Medición (fórmula)	X = A / B A= número de funciones de diagnóstico tal como se especifica confirmado en la revisión. B = número de funciones de diagnóstico necesarias.
Interpretación del valor obtenido	0 <= X La ejecución más cercana a 1, la mejor de las funciones de diagnóstico.

Métrica de cambiabilidad	
Nombre de la métrica	Grabación de cambio
La métrica se propone medir	¿Están los cambios en las especificaciones y los módulos del programa registrado adecuados a las líneas de comentarios en el código?
Método de aplicación	Proporción del registro de información de cambio de módulo.
Medición (fórmula)	X= A/B A = número de cambios en la función de los módulos con un cambio de comentario confirmado en la revisión. B = número total de la función de los módulos que ha cambiado desde el código original.
Interpretación del valor obtenido	0 <= X <= 1 Cuanto más se acerque a 1, mayor grabación.

Métrica de estabilidad	
Nombre de la métrica	Impacto de cambio
La métrica se propone medir	¿Cuál es la frecuencia de efectos adversos después de la modificación?
Método de aplicación	Cuente el número de efectos adversos a detectar después de la modificación y la comparamos con el número de modificaciones realizadas.
Medición (fórmula)	X= 1- A/B A = número de efectos adversos a detectar después de las modificación. B = número de modificaciones realizadas.
Interpretación del valor obtenido	0 <= X <= 1 Cuanto más se acerque a 1, mejor.

Métrica de capacidad de prueba	
Nombre de la métrica	Autonomía de la capacidad de prueba
La métrica se propone medir	¿Cómo puede el software de forma independiente ser probado?
Método de aplicación	Cuente el número de dependencias en otros sistemas de pruebas que se han simulado con colillas y compararlo con el número total de las dependencias de la prueba en otros sistemas.
Medición (fórmula)	X= A/B A = número de dependencias en otros sistemas de pruebas que se han simulado con colillas. B = número total de las dependencias de la prueba en otros sistemas.
Interpretación del valor obtenido	0 <= X <= 1 Cuanto más se acerque a 1, mejor.

Métrica de cumplimiento de la mantenibilidad	
Nombre de la métrica	Cumplimiento de la mantenibilidad
La métrica se propone medir	¿Cómo es compatible el mantenimiento del producto con las normas aplicables, las normas y convenciones?
Método de aplicación	Cuente el número de requisitos que requieren de cumplimiento que se han cumplido y comparar con el número de requisitos que requieran el cumplimiento.
Medición (fórmula)	X= A/B A = número de requisitos correctamente relacionados con el cumplimiento de mantenimiento. B= número total de elementos de cumplimiento.
Interpretación del valor obtenido	0 <= X <= 1 Cuanto más se acerque a 1, el más dócil.

GLOSARIO

Cliente: Una persona u organización, interna o externa a la organización productora que toma responsabilidad financiera por el sistema. El cliente es el último destinatario del producto desarrollado y sus artefactos.

Producto de Software: (IEEE-12207) Es el conjunto de programas de computadora, procedimientos, documentación y datos, asociados.

Proceso: (ISO-15504) Proceso de Software, es el proceso (o procesos), usado por una organización (o proyecto), para planificar, administrar, ejecutar, monitorear, controlar y mejorar sus actividades, relacionadas con el software.

ISO: Es la Organización Internacional para la Normalización (del inglés International Organization for Standardization); responsable para la normalización a escala mundial. ISO está formado por distintos comités técnicos, cada uno de los cuales es responsable de la normalización para cada área de especialidad. El propósito de ISO es promover el desarrollo de la normalización para fomentar a nivel internacional el intercambio de bienes y servicios y para el desarrollo de la cooperación en actividades económicas, intelectuales, científicas y tecnológicas. El resultado del trabajo técnico dentro de ISO se publica en forma final como normas internacionales.

Mantenibilidad: característica (requisito no funcional) de calidad que debe poseer el software, permitiendo que este sea capaz de ser modificado para evolucionar ante la inclusión de nuevos requerimientos o necesidades de cambio de ambiente.