UNIVERSIDAD INTERNACIONAL SEK

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

Trabajo de fin de carrera titulado:

ESTUDIO DE LAS BASES CONCEPTUALES, PARA LA IMPLEMENTACIÓN DE LAS MEJORES PRÁCTICAS PARA EL CONTROL Y ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE.

Realizado por:
DIEGO PATRICIO VINUEZA MOREJÓN

Como requisito para la obtención del título de:
INGENIERO EN SISTEMAS INFORMÁTICOS

QUITO, ABRIL DE 2009

DECLARACIÓN JURAMENTADA

Yo, Diego Patricio Vinueza Morejón, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

DIEGO PATRICIO VINUEZA MOREJÓN

DECLARATORIA

El presente trabajo de investigación de fin de carrera, titulado

ESTUDIO DE LAS BASES CONCEPTUALES, PARA LA IMPLEMENTACIÓN DE LAS MEJORES PRÁCTICAS PARA EL CONTROL Y ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

Realizado por el alumno:

DIEGO PATRICIO VINUEZA MOREJÓN

Como requisito para la obtención del título de:

INGENIERO DE SISTEMAS INFORMÁTICOS

Ha sido dirigido por el profesor:

Ing. VICENTE GARCÉS

Quien considera que constituye un trabajo original de su autor.

…….

Ing. VICENTE GARCÉS
Director

Los profesores informantes

Ing. José Sancho

Ing. Santiago Mena

Después de revisar el trabajo escrito presentado,

lo han calificado como apto para su defensa oral ante el tribunal examinador

……..…… ……..……

Ing. José Sancho Ing. Santiago Mena

Quito, a 20 de Abril de 2009

DEDICATORIA

A mi madre la Sra. Alcira Morejón de Vinueza, por ser la luz de mis ojos la persona que guía mis pasos, mi ángel, mi confidente.

A mi padre el Arq. Patricio Vinueza por su lucha constante, claro ejemplo de persona honesta, amable y solidaria, mil gracias por su entrega y trabajo diario, usted y yo sabemos que este triunfo es fruto de una dura decisión que marco nuestras vidas, estaré eternamente agradecido porque gracias a usted soy lo que soy.

A mi hermano Juan Sebastián Vinueza por su compañía y su buen humor, quien siempre tenía una palabra para animarme a seguir luchando y asumir los retos del día a día.

A mi familia, abuelos, tíos primos por su cariño y constante apoyo.

AGRADECIMIENTO

A Dios por darme la vida.
A mis padres, porque creyeron en mi y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mí, fue lo que me hizo ir hasta el final.
Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

A la empresa Intergrupo S.A y a su gerente general el Ing. Boris Arciniegas por brindarme la oportunidad y permitirme ser parte de su equipo de trabajo.

A mis compañeros de trabajo, excelentes profesionales, ilustres personas que a diario gracias a sus enseñanzas alimentaron mis conocimientos, mil gracias muchachos.

Un agradecimiento especial a mi director de tesis el Ing. Vicente Garcés, excelente maestro quien, fue una guía y pilar fundamental en el desarrollo de este documento.

Mi agradecimiento para la Universidad Internacional SEK y a la facultad de Sistemas encabezada por la Ing. Viviana Guerrón excelente decana y maestra de quien recibí la preparación y los conocimientos para enfrentar este nuevo reto en mi vida.

En general a todos aquellos que me apoyaron y estuvieron preocupados por mí, por mi trabajo y por el desarrollo de este documento.
Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida, espero no defraudarlos y contar siempre con su valioso, sincero e incondicional apoyo.

RESUMEN

En la actualidad las labores de aseguramiento de calidad del software se han convertido en un factor común y cada vez más importante entre las diversas entidades (empresas, organizaciones, personas desarrolladoras de software) que utilizan y desarrollan aplicaciones de software.

Con frecuencia las tareas que deben cumplir las aplicaciones de software son catalogadas como críticas provocando en la actualidad una creciente dependencia de los sistemas computacionales, donde alguna falla puede resultar en catástrofes personales (sistemas informáticos críticos) o pérdidas económicas (sistemas informáticos no críticos).

Pese a que se conoce la necesidad de producir software de calidad, la cultura actual de la calidad enseña, que en las organizaciones los administradores empiezan a involucrarse en los procesos de desarrollo de tecnología una vez que se han incurrido en costos de mantenimiento, ya sean ocasionados por un mal diseño, o por no satisfacer los requerimientos correctamente

Con este contexto, la calidad del software es una factor determinante para lograr sistemas informáticos confiables y estables, considerando que la calidad en el desarrollo de Software es una actividad que ha surgido como consecuencia de la fuerte demanda de Sistemas de Software que se desarrollan en estos días; que van desde programas elementales de contabilidad hasta programas complejos como los sistemas de control de tráfico aéreo en los aeropuertos, sistemas bancarios o sistemas de monitoreo medico.

De esta manera la calidad del software ha dejado de ser una opción y se ha convertido en una obligación para las diversas entidades que interactúan con el software garantizando entre otras cosas, por medio del aseguramiento de calidad, la disponibilidad del negocio.

ABSTRACT

It’s common nowadays that the use of software quality assurance techniques has become a common task that takes an increasing importance in different kinds of companies which use and develop software applications.

Frequently the tasks that must be provided by software applications are considered critical causing an increase on the dependency of computer systems, where any failure can cause major human catastrophes (critical systems) and economical catastrophes (non-critical systems)

Although the need for quality software has been identified, our current culture demonstrates that in organizations, managers only start to involve in technology development processes once a maintenance cost has been identified, be this originated by a bad design, or because the original requirements weren’t satisfied correctly.

In this context, software quality assurance is a determining factor to obtain trustworthy and stable systems, considering that software quality assurance is an activity that developed as a consequence of a strong demand of Software Applications which range from simple accounting applications to the most complex air traffic control systems, banking systems or even medical monitoring software.

In this way, quality software is no longer an option and has become a requirement for any entity that interacts with the application, guaranteeing among other things, the business's continuous availability.

TABLA DE CONTENIDO

LISTADO DE TABLAS Y FIGURAS

CAPÍTULO 1.- DISEÑO CONCEPTUAL DE LA INVESTIGACIÓN

1.1.DETERMINACIÓN DEL PROBLEMA

En la actualidad las labores de aseguramiento de calidad del software se han convertido en un factor común y cada vez mas importante entre las diversas entidades (empresas, organizaciones personas desarrolladoras de software) que utilizan y desarrollan aplicaciones de software.
Pero por qué es tan importante y casi imprescindible la calidad del software? Es muy importante ya que permite consolidar sistemas informáticos que garanticen la operación confiable del negocio, evitando interrupciones no planificadas que deterioren la imagen empresarial.
La mayoría de veces estas fallas ocasionan grandes impactos económicos que, en muchos casos, representan perdidas lamentables de dinero.

Con este contexto, la calidad del software es una factor determinante para lograr sistemas informáticos confiables y estables, considerando que la calidad en el desarrollo de Software es una actividad que ha surgido como consecuencia de la fuerte demanda de Sistemas de Software que se desarrollan en la actualidad; que van desde programas elementales de contabilidad hasta programas complejos como los especialistas. De allí el esfuerzo que se ha desplegado para obtener software de alta calidad.
De esta manera la calidad del software ha dejado de ser una opción y se ha convertido en una obligación para las diversas entidades que interactúan con el software garantizando entre otras cosas, por medio del aseguramiento de calidad, la disponibilidad del negocio.

El desarrollo de software se ha convertido en una actividad programada, planificada y con un ciclo de vida bien definido que le otorga un carácter formal.
El concepto de la IEEE 1074 define al ciclo de vida del software como: "Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software"… [1].
Mientras para ISO 12207-1 el concepto de ciclo de vida es "Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de requisitos hasta la finalización de su uso"…[2]
El ciclo de vida del software involucra los procesos de desarrollo y de soporte.
Los Procesos de desarrollo involucra: el análisis de los requisitos del software, el diseño detallado del software, la codificación y las pruebas del software.
Mientras que los procesos de soporte incluye: El aseguramiento de la calidad, el proceso de verificación y el proceso de validación.

[1] [MONTILVA. Jonás]
[2] [RAGHU. Singh]

Con frecuencia, en los proyectos se suele prescindir del proceso de aseguramiento de calidad, ante presiones de tipo económicas o de tiempo, alegando que es una actividad no remunerada por el cliente y que no es realmente necesaria su aplicación, siempre y cuando el producto de software funcione correctamente.

Históricamente, lo que se hacía era limitar el proyecto a la producción de código y, en la etapa final del proyecto se aplicaba una fase corta de verificación para verificar que el sistema funciona correctamente.
Este accionar no era para nada conveniente ya que, de existir un defecto o falla, el costo de la reparación del mismo resultaba ser extremadamente altos en términos económicos y de tiempo.
De esta manera se prescindía de las actividades "extras" o de soporte que finalmente lo que buscan es asegurar una calidad en el producto, que el producto sea utilizado por el cliente y perdurable en el tiempo. Es por esto que algunos desarrolladores creen que la calidad del software es algo en lo que deben preocuparse luego de haber generado el código.
Pues no es así ya que el aseguramiento de calidad del software es una actividad de prevención, que se aplica a lo largo de todo ciclo de vida del software, dedicando un mayor esfuerzo al momento de ejecutar el proceso de soporte, y así cumplir con el ciclo de vida ya definido.

Uno de los aspectos más descuidados en este gran proceso de desarrollo de software es el de asegurar la calidad, ya que constituyen la herramienta y el medio por el cual se puede asegurar la calidad del producto. Cumplir con los requisitos del usuario en la mayoría de los casos no es suficiente, para ello hace falta definir un proceso de pruebas serio y aplicable para todos los proyectos, es decir, estandarizado y que incluya las mejores prácticas de pruebas siempre bajo un modelo que asegure la mejora continua.

Una alternativa para alcanzar competitividad en la industria del software requiere desarrollar y aplicar un modelo basado en metodologías o procedimientos estándares para el planeamiento, especificación y ejecución de pruebas de software.

En la mayoría de universidades que tienen la carrera Ingeniería Informática, las actividades de pruebas de software pasan casi desapercibidas ya que no se crean casos de pruebas que permitan garantizar la calidad del software.
La ausencia de una orientación clara en la planificación del proyecto y de políticas organizacionales que apoyen este proceso debido al desconocimiento o inaplicabilidad de algunos modelos de calidad aumentan el riesgo de producir software de muy mala calidad.
No se trata sólo de invertir más tiempo o contratar más personal, para la ejecución de pruebas. De nada sirve poner a trabajar "*testers*" (probadores de aplicaciones) en el sistema que se está desarrollando, si no se cuenta con una metodología que defina un marco de trabajo con actividades y responsabilidades sobre el cual se va a trabajar para así asegurar la calidad del software.

El presente trabajo no será una definición estricta de una metodología para asegurar la calidad del software, sino más bien una recopilación completa y detallada respecto a lo que son los sistemas, los modelos de calidad y la conceptualización correcta de términos para garantizar la calidad del software.

Este trabajo pretende ser un conjunto de lineamientos y conocimientos de importancia, producto de una investigación ardua, que puede servir de base para la estructuración de un sistema de control de calidad de software que las entidades de desarrollo de software, hoy en día, deberían tomar en cuenta con la finalidad de asegurar la calidad en sus productos de software y así lograr ser más competitivos en sus negocios.

1.2.DEFINICIÓN DEL TEMA

Estudio de las bases conceptuales para la implementación de las mejores prácticas para el control y aseguramiento de la calidad del software.

1.3.JUSTIFICACIÓN E IMPORTANCIA

Esta investigación es necesaria para las entidades que interactúan con el software tales como: instituciones educativas, empresas públicas y privadas, municipalidades, entre otras; ya que éstas procesan grandes cantidades de información, que es considerada como un activo importante y vital para su operación.
Debido a que esta información crece día a día, es fundamental que dichas empresas se preocupen por buscar la calidad en cada una de sus aplicaciones de tal manera que su negocio no se vea afectado por el mal funcionamiento de cualquiera de sus aplicativos de software. De esta manera estas entidades exigirán un producto de software que ha pasado por el proceso de aseguramiento de calidad, y que se garantice una alta calidad, y un mejor desempeño del software en producción evitando la presencia de fallas o errores que puedan dificultar y afectar la continuidad del negocio ocasionando grandes pérdidas de inversión.

Asimismo, el trabajo es conveniente para las entidades desarrolladoras de software; porque tienen la necesidad de asegurar la calidad del software que desarrollan, con trabajos como este se intenta impulsar a las entidades desarrolladoras a poner más énfasis a las actividades de aseguramiento de Calidad.

Definitivamente la Calidad del software es vital, tanto para entidades consumidoras de software como para las entidades que lo desarrollan. Es por esto que la importancia del tema a investigar está relacionado con un problema actual: Contar con productos de software libre de fallas y errores.
De esta manera la intención de este trabajo es rescatar, y entender los conceptos involucrados en el aseguramiento de la calidad del software, así como conocer los lineamientos de calidad en los que se basan las metodologías para conseguir que los productos de software cuenten con los estándares que ayuden a dotar de calidad al software.

1.4.OBJETIVOS

1.4.1. General

Generar una guía de conceptos y lineamientos bases para asegurar la calidad del software de tal manera que las entidades tengan, en este documento, un punto de partida para entender qué es y cómo funciona la calidad del Software y puedan utilizar esta guía para implementar y cumplir con los requerimientos básicos y lograr que las aplicaciones que se desarrollen sean tomadas en cuenta como un producto realizado con calidad, en caso de ser entidades desarrolladoras de software. En caso de ser una entidad consumidora de software, que la misma se asegure que el producto que le están entregando cumpla con la calidad esperada.

1.4.2. Específicos:

- Conocer en qué consiste la Garantía de Calidad en un proyecto de desarrollo de Software.
- Conocer los puntos básicos que se debe tener en cuenta a la hora de construir Software bajo un Sistema de Garantía de Calidad.
- Conocer los tipos de Pruebas que se ejecutan en el desarrollo de un plan de pruebas.
- Dotar al personal de Aseguramiento de calidad del conocimiento necesario para que pueda ser capaz de planificar y efectuar las pruebas del software que se desarrolle.
- Estudiar sobre normas y principios de calidad del software
- Conocer qué tipo de herramientas pueden ayudar a la automatización de pruebas del software
- Comprender la importancia de la calidad en el proceso de desarrollo de software.
- Conocer los costes que implica la calidad, y la no calidad en los productos que se desarrollan
- Comprender que la mejora en la calidad del software es posible comprobar a través de la medición, evaluación y la aplicación de técnicas, métodos y herramientas asociadas con el proceso de desarrollo.
- Analizar las metodologías particulares aplicadas a los procesos de pruebas de software.
- Justificar la importancia del proceso de pruebas en el ciclo de vida del software, como un medio de aseguramiento de calidad.

1.5.HIPÓTESIS

El presente trabajo no pretende comprobar ninguna hipótesis sino más bien en base a una investigación detallada reunir un conjunto mejores prácticas para el control y el aseguramiento de calidad del software.

1.6. DELIMITACIÓN DEL TEMA

La presente investigación se limitará a definir la importancia del aseguramiento de la calidad en el software. Se presentarán conceptos previos para ayudar a aclarar los temas relacionados, y justificar la investigación; más adelante se analizará el proceso de aseguramiento de calidad del software como medio para garantizar la calidad por excelencia, este trabajo también propone realizar un breve estudio de los estándares y modelos de madurez más usados ISO, CMMI, SIX SIGMA, SPICE

1.7. LIMITACIÓN DE LA INVESTIGACIÓN

El aseguramiento de calidad del software en nuestro país es una actividad que en estos días no es muy conocida. Las empresas de desarrollo de software en el Ecuador apenas están implementando este servicio, es por esto que el presente trabajo pretende ser considerado por estas empresas de desarrollo de software como una guía que puede servir para el establecimiento de un verdadero sistema de control de calidad del software.
En el Ecuador el aseguramiento de calidad del software no está totalmente constituido, por este motivo no será posible contar con fuentes de investigación locales. Sin embargo se ha realizado un estudio de investigación seleccionando empresas en la ciudad de Quito, que se encuentran involucradas con el software, la selección de las empresas se realizo sin algún orden especifico ni preferencia alguna, de tal manera que utilizaremos una encuesta la misma que nos servirá para determinar y conocer que tan involucrados están dichas empresas en el proceso de aseguramiento de calidad del Software.
Cabe destacar también que para el desarrollo de este trabajo se han tomado en cuenta fuentes de investigación provenientes de otros países como por ejemplo: Estados Unidos o Europa donde esta actividad ha dejado de ser un servicio y se ha convertido en un factor importante en el proceso de desarrollo de Software.

1.8. MARCO TEÓRICO

1.8.1. Ciclo de vida clásico del desarrollo de sistemas

El ciclo de vida para el desarrollo de sistemas es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información. El método del ciclo de vida para el desarrollo de sistemas consta de 6 fases:

1). Investigación Preliminar: La solicitud para recibir ayuda de un sistema de información puede originarse por varias razones: sin importar cuales sean estas, el proceso se inicia siempre con la petición de una persona.

2). Determinación de los requerimientos del sistema: El aspecto fundamental del análisis de sistemas es comprender todas las facetas importantes de la parte de la empresa que se encuentra bajo estudio. Los analistas, al trabajar con los empleados y administradores, deben estudiar los procesos de una empresa para dar respuesta a las siguientes preguntas clave:

¿Qué es lo que hace?, ¿Cómo se hace?, ¿Con que frecuencia se presenta?, ¿Qué tan grande es el volumen de transacciones o decisiones?, ¿Cuál es el grado de eficiencia con el que se efectúan las tareas?, ¿Existe algún problema? ¿Qué tan serio es? ¿Cuál es la causa que lo origina?

3). Diseño del sistema: El diseño de un sistema de información produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. Los especialistas en sistemas se refieren, con frecuencia, a esta etapa como diseño lógico en contraste con la del desarrollo del software, a la que denominan diseño físico.

4). Desarrollo del software: Los encargados de desarrollar software pueden instalar software comprobando a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para escribir el software y de la disponibilidad de los programadores.

Por lo general, los programadores que trabajan en las grandes organizaciones pertenecen a un grupo permanente de profesionales.

5). Prueba de sistemas: Durante la prueba de sistemas, el sistema se emplea de manera experimental para asegurarse de que el software no tenga fallas, es decir, que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga.

Se alimentan como entradas conjunto de datos de prueba para su procesamiento y después se examinan los resultados.

6). Implantación y evaluación: La implantación es el proceso de verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla. Una vez instaladas, las aplicaciones se emplean durante muchos años. Sin embargo, las organizaciones y los usuarios cambian con el paso del tiempo, incluso el ambiente es diferente con el paso de las semanas y los meses.

Por consiguiente, es indudable que debe darse mantenimiento a las aplicaciones.

1.8.2. Concepto de calidad

1.8.2.1. ¿Qué es la calidad del software?

La calidad está de moda, en todos los aspectos, pero especialmente en el desarrollo de software. El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. Ahora bien, ¿qué es la calidad del software?
A la hora de definir la calidad del software se pueden adoptar diferentes aproximaciones.
Como primera aproximación es importante diferenciar entre la calidad del PRODUCTO de software y la calidad del PROCESO de desarrollo. No obstante, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la

calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

También es importante destacar que la calidad de un producto software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, código). No basta con tener en cuenta la calidad del producto una vez finalizado, cuando los problemas de mala calidad ya no tienen solución o la solución es muy costosa.

1.8.2.2. Principios considerados para que un Software sea de calidad.

- Que sea de fácil uso para el usuario, es decir que el usuario no tenga que ejecutar procesos complicados, para poder obtener lo que desea. El principio de usabilidad debe estar presente para que el software sea considerado de calidad.
- Que cumpla con lo especificado y solicitado por el cliente.
- La calidad de fabricación es la fidelidad con que un producto se ajusta a lo establecido en su proyecto, o sea como se apega a las necesidades y requerimientos del cliente

1.8.2.3. Impacto de una aplicación de software sin Control de calidad

- Incremento en los costos del desarrollo
- Retrasos en los cronogramas
- Desborde de presupuestos
- Flujos de trabajo interrumpidos y frustración por parte de los clientes
- Despliegue de soluciones incompletas
- Los sistemas que no cuentan con los mínimos requerimientos de calidad generan desconfianza e insatisfacción por parte del usuario

1.8.2.4. Objetivo del Aseguramiento de calidad

Asegurar la aceptación de un Sistema, a través de la verificación del cumplimiento de todos los requerimientos especificados y acordados, así como también el cumplimiento de los estándares de presentación y uso de la aplicación, establecidos por la compañía.

1.8.2.5. Que son las pruebas del Software[3]

La prueba es un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática.
Una vez generado el código fuente el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente.

1.8.2.6. Características de las pruebas de software[3]

Se deben realizar revisiones técnicas formales y efectivas. Esto eliminará muchos errores antes de que empiece la prueba.
La prueba comienza a nivel de componentes y trabaja hacia la integración de todo el sistema

1.8.2.7. Proceso de V&V (Verificación y validación)[3]

Verificación y validación (V&V) es el nombre dado a los procesos de análisis y pruebas estos tienen lugar en cada etapa del proceso del software
Comienzan con la revisión de los requerimientos, continúa con revisiones del diseño, inspecciones del código, hasta la prueba del producto.
Verificación: es el conjunto de actividades que aseguran que el software implemente correctamente una función específica. ¿Estamos construyendo el producto correctamente?

Validación: es un conjunto diferente de actividades que aseguran que el software construido corresponde con los requisitos del cliente. ¿Estamos construyendo el producto correcto?

1.8.2.8. ¿Cuando se termina la etapa de pruebas?

El límite de las pruebas de software se lo observa de dos diferentes enfoques: el primero de ellos dice que: Nunca se termina de aplicar una prueba; la carga simplemente se desplaza de los desarrolladores a los clientes.
Otro de los enfoques con el que se estudia la etapa de pruebas es que: La prueba se termina cuando se agota el tiempo o el dinero.
No se puede estar seguro de que el software no fallará, pero se puede dar un valor estadístico teóricamente sólido y validado de forma experimental de que existe una pequeña probabilidad de fallo en determinado tiempo de uso

3 [GARCES.V]

1.8.2.9. Como implementar un plan exitoso para Asegurar la calidad de los productos de software

Si se desea implementar con éxito un plan QA de software se deben atender los siguientes temas:

- Especificar los requisitos del producto de manera cuantificable mucho antes de que empiecen las pruebas
- Establecer explícitamente los objetivos de la prueba
- Comprender cuales son los usuarios del software y desarrollar un perfil para cada categoría de usuario
- Construir un software robusto diseñado para probarse a si mismo
- Usar técnicas y revisiones formales y efectivas como filtro previo a la prueba
- Mejora continua para el proceso de prueba.

1.9. DISEÑO DE LA INVESTIGACIÓN

Se ha diseñado el presente trabajo bajo una investigación documental, el estudio de problemas con el propósito de ampliar y profundizar el conocimiento referente al aseguramiento de calidad del Software, basado principalmente, en publicaciones, libros y fuentes de información publicados en medios impresos, audiovisuales y electrónicos.

La investigación documental es una técnica que permite obtener documentos nuevos en los que es posible describir, explicar, analizar, comparar, criticar sobre un tema o asunto mediante el análisis de fuentes de información.

La investigación está basada también en las vivencias diarias del autor del presente documento ya que cuenta con experiencia en el aseguramiento de calidad del software.

La originalidad del texto se reflejará en el enfoque, criterios, conceptualizaciones, reflexiones, conclusiones, recomendaciones y, en general, en el pensamiento del autor.

CAPÍTULO 2.- DEFINICIÓN DE CALIDAD DEL SOFTWARE

Este capítulo está orientado a identificar y definir conceptos claves que serán utilizados a lo largo de esta investigación.

2.1.DEFINICIÓN DE CALIDAD.

El término calidad puede resultar ambiguo e incluso subjetivo porque, como la belleza, la calidad depende de quien la observa.
Es por esto que es necesario definir el concepto de Calidad con claridad, ya que si la calidad no puede ser definida, no puede ser medida; y donde la calidad no puede ser medida entonces no puede ser controlada.

La calidad se define como "grado en que un conjunto de características inherentes cumple con unos requisitos"[4].

El Aseguramiento de la Calidad pretende dar confianza, asegurando que el producto reúne las características necesarias para satisfacer todos los requisitos solicitados por el cliente.

De esta manera, para asegurar la calidad de los productos resultantes el equipo de
Calidad deberá realizar un conjunto de actividades que servirán principalmente para, reducir, eliminar y lo más importante, prevenir las deficiencias de calidad del o los productos que se van a obtener.

2.2.¿QUÉ ES LA CALIDAD DEL SOFTWARE?

"…La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. …"[5]

La calidad del software es medible y varía de un sistema a otro. Un software elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de software para ser explotado durante un largo período, necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación

[4] [ISO 9000:2000]
[5] [FERNANDEZ. O]

2.3. EVOLUCIÓN DE LA CALIDAD Y LA CALIDAD DEL SOFTWARE

2.3.1. Etapa Primera. Desde la revolución industrial hasta 1930.

El nacimiento de la Calidad surge a raíz de la revolución industrial etapa que significó la transición del trabajo manual por el trabajo mecánico.
Antes de esta etapa el trabajo se realizaba de forma artesanal, y la principal característica era que el trabajador era el responsable de la producción completa de un producto.

A inicios de 1900 aparece la figura del supervisor, quien en muchas ocasiones era el mismo propietario
Durante la Primera Guerra Mundial, los sistemas de fabricación se hicieron más complicados y como resultado de esto aparecen los primeros inspectores de calidad a tiempo completo, esto provocó que se realizara una división del área de producción en dos aéreas, producción como tal e inspección.

Esta época se caracterizó por la inspección, cuyo principal interés era la detección de los productos defectuosos para separarlos de los productos aptos para la venta.

2.3.2. Segunda Etapa. 1930-1949.

Cuando surgió la Segunda Guerra Mundial, las necesidades de la enorme producción en masa requirieron del control estadístico de la calidad.

Las labores de inspección eran demasiado estrictas pero un control estadístico de la calidad provoco una nueva técnica llamada inspección por muestreo, en lugar de una inspección al cien por ciento
Hasta este tiempo los intereses de esta época eran el conocer y seleccionar los desperfectos o los productos con fallas. En esta etapa surgió también la toma de acción correctiva sobre los procesos tecnológicos.

Sin lugar a duda los inspectores de la calidad eran un elemento clave en el resultado de la producción de la empresa, de tal manera que su actividad aumento y ya no solo realizaban la inspección del producto en su instancia final, su labor se despliega a lo largo de todo el proceso de producción

2.3.3. Tercera Etapa. 1950-1979.

Luego de la segunda guerra mundial la calidad se mantiene fuerte con la inspección, tratando de no sacar a la venta productos defectuosos.
Al poco tiempo surge y gracias a la inspección a lo largo del proceso se llega a la conclusión de que el problema de los productos defectuosos iniciaba en las diferentes fases del proceso

Debido a esto se pasa de la inspección al control de todos los factores que intervienen en el proceso desde la etapa inicial hasta la satisfacción final de todos los requisitos del consumidor
Luego de algunos análisis se llegó a la conclusión de que éste era el enfoque correcto y el interés principal puso énfasis en la coordinación de todas las aéreas organizativas en función del objetivo de lograr la calidad en los productos.

El asunto de la calidad venía evolucionando, es por esto que aparecieron programas y se desarrollaron sistemas de calidad para las áreas de calidad de las empresas, donde además de la medición de la calidad, se incorpora la planeación de la calidad. Con un enfoque de que la calidad se construye desde adentro
En Japón se empieza a escuchar las primeras conferencias del Dr. Edwards Deming, marcando el inicio del cambio de Cultura hacía la calidad preventiva.

El Dr. Edwards Deming difusor y promotor de la calidad total, enmarca su teoría en 14 puntos muy importantes:

1. Ser constante en el propósito de mejorar los productos y los servicios.
2. Adaptar la nueva filosofía.
3. No depender más de la inspección Masiva
4. Acabar con la práctica de adjudicar contratos de compra basándose exclusivamente en el precio.
5. Mejorar continuamente y por siempre el sistema de producción y de servicio.
6. Instituir la capacitación en el trabajo.
7. Instituir el liderazgo.
8. Desterrar el Temor.
9. Derribar las barrearas que existan entre áreas del Staff.
10. Eliminar Slogans las exhortaciones y las metas para la fuerza laboral.
11. Eliminar las cuotas numéricas.
12. Derribar las barreas que impiden el sentimiento de orgullo que produce el Trabajo bien hecho.
13. Establecer un vigoroso programa de educación y de reentrenamiento.
14. Tomar medidas para logra la transformación

El Dr. Deming dice que "*si una gran masa crítica de la compañía entiende y practica los 14 puntos será posible manejar las siete enfermedades"* que se presentan a continuación:

1. Falta de constancia en el propósito.
2. Énfasis en las utilidades a corto Plazo.
3. Evaluación de desempeño, clasificación según su logro anual.
4. La movilidad de la Gerencia.
5. Manejar la compañía basándose únicamente en cifras visibles.
6. Costos médicos excesivos.
7. Costos excesivos de garantía fomentados por abogados que cobran con base en honorarios altos para los casos imprevistos de demandas.

Gracias a los inicios de la teoría de calidad total del Dr. Deming en Japón aparecen también conceptos como el Círculo de calidad y el JIT (Just In Time).
A continuación se presenta una breve descripción de los conceptos antes mencionados:

Circulo de la calidad:
Un Círculo de Calidad está integrado por los empleados de la misma área de trabajo y su supervisor, que se reúnen voluntaria y regularmente para estudiar técnicas de mejoramiento de control de calidad y de productividad, con el fin de aplicarlas en la identificación y solución de dificultades relacionadas con problemas vinculados a sus trabajos. El círculo de calidad tiene por objetivo crear conciencia de calidad y productividad en todos y cada uno de los miembros de una organización, a través del trabajo en equipo y el intercambio de experiencias y conocimientos, así como el apoyo recíproco.

Just In Time

El JIT es una filosofía de producción que se orienta a la demanda. La ventaja competitiva ganada deriva de la capacidad que adquiere la empresa para entregar al mercado el producto solicitado, en un tiempo breve, en la cantidad requerida.

La idea de JIT es eliminar el despilfarro, lo que obliga a eliminar todos los insumos de recursos que no añaden valor al producto o servicio, todo esto para proporcionar satisfacción al cliente al tiempo de minimizar los costos de producción

2.3.4. Cuarta Etapa. Década del 80.

La parte principal de esta etapa no era solo el mercado de manera general sino el conocimiento de las necesidades del cliente, para crear productos que satisfagan las necesidades del cliente.

Se definió que la responsabilidad de la calidad es en primer lugar de la alta dirección, sin dejar de pensar que para lograr calidad, deben participar todos los miembros de la organización.
Para esta etapa la calidad era vista como una oportunidad competitiva.

2.3.5. Quinta Etapa. 1990 hasta la fecha.

Esta etapa se conoce como Servicio de Calidad Total.
Los clientes actuales solo están dispuestos a pagar por lo que significa valor para ellos por esto es que la calidad es apreciada por el cliente desde dos puntos de vista, calidad perceptible y calidad factual.
Calidad perceptible es lo que lleva a los clientes a adquirir el producto, calidad factual es la responsable de crear y lograr la lealtad del cliente con la marca.

2.4. ASEGURAMIENTO DE CALIDAD DE SOFTWARE (SQA)

El aseguramiento de calidad tiene como actividad primaria el determinar si las necesidades de los usuarios están siendo satisfechas adecuadamente.
Otra de sus funciones, importante también, es la de determinar los costos que puede causar el añadir ciertas características al producto, ya que tarde o temprano, la economía resulta ser un factor decisivo para obtener un producto de calidad.

"Para determinar si las necesidades de los usuarios están siendo satisfechas, se deben de evaluar tres áreas:

- Objetivos: Los objetivos de la organización son primero, luego vienen los requerimientos del usuario. Los objetivos de cualquier usuario deben de estar en armonía con los objetivos de la organización.

- Métodos: Deben de utilizarse métodos que contengan u observen las políticas, procedimientos y estándares de la organización,

- Ejecución: Optimización del uso de hardware y software al implementar los productos de software"[6]

La forma más optima y real de asegurar la calidad de un producto de software es evaluando las tres áreas antes mencionadas, para esto es necesario que exista un programa de aseguramiento de calidad que sea efectivo y que tenga un impacto dentro del desarrollo y prueba del producto de software final.

2.5. NECESIDAD DE LA CALIDAD Y DE SUS PROCESOS DE ASEGURAMIENTO.

Las organizaciones que desarrollan software en la actualidad, se encuentran en una situación donde deben idear estrategias que las pongan en ventaja con sus inmediatos competidores y las Tic`s (tecnologías de información) son herramientas usualmente escogidas con este propósito.

Lo antes mencionado queda sustentado con los siguientes apartados que se describen a continuación:
"La naturaleza crítica de algunas tareas realizadas por las computadoras. En la actualidad se está dando una creciente dependencia de los sistemas computacionales, donde alguna falla puede resultar en catástrofes personales (sistemas de control aéreo, en los aeropuertos) y económicas (sistemas transaccionales en los bancos).

El crecimiento de los costos de desarrollo de productos de software. Los costos causados por mantenimiento de software son cada vez mayores, por lo que se vuelve indispensable evitar errores desde la definición de requerimientos"[7].

[6] [PERRY, WILLIAM E]
[7] [LITTLEWOOD B and N. FENTON]

"Pese a que se conoce la necesidad de producir software de calidad, la cultura actual de la calidad enseña que en las organizaciones los administradores empiezan a involucrarse en los procesos de desarrollo de tecnología de información una vez que se han incurrido en costos de mantenimiento, ya sean ocasionados por un mal diseño, o por no satisfacer los requerimientos correctamente"[8].

Como se ha mencionado en los apartados citados se destaca la importancia de la calidad del software debido a que en la actualidad las acciones consideradas como criticas en cualquier ámbito están siendo ejecutadas por computadoras y por sistemas de información.

2.6. BENEFICIOS DE LOS PROCESOS DE ASEGURAMIENTO DE LA CALIDAD EN EL SOFTWARE.

Los beneficios que se pueden obtener como resultado de aplicar los procesos de aseguramiento de calidad son muchos y variados, algunos que se pueden citar con brevedad son:

1. Se detectan problemas rápidamente, Es posible identificar problemas en tempranas etapas del desarrollo de productos de software, ayudando al desarrollador a corregirlos inmediatamente y poder avanzar con más rapidez.
Esta comprobado que el detectar un defecto en las etapas tempranas del ciclo del desarrollo es económicamente menos costoso y menos riesgoso que el corregirlos en las etapas finales del proceso de desarrollo de software.

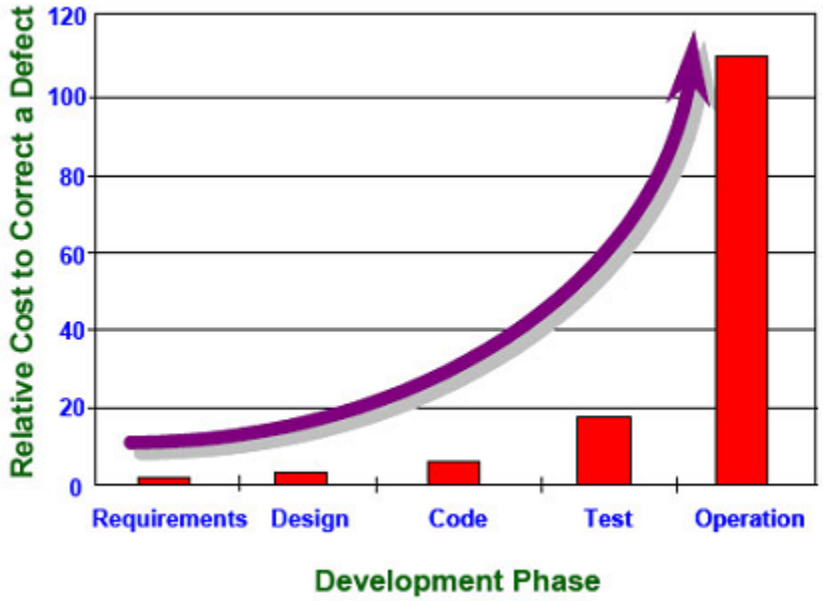


Figura 001.- Costo de la corrección de errores [9]

[8] [PERRY, WILLIAM E]
[9] [BOEHM, B]

2. Se crean y se siguen estándares de trabajo, Con apoyo del proceso de aseguramiento de calidad, se pueden establecer estándares tan diversos como son los de codificación o de documentación, los cuales apoyan a uniformizar y consolidar el proceso de desarrollo.

3. Se verifica que los objetivos individuales vayan acordes con los objetivos de la organización, Se busca y se recomienda que los requerimientos expuestos por usuarios finales estén alineados con los objetivos globales de la empresa, facilitando así el logro de los mismos y la integración total de los usuarios a la organización.

4. Se recomiendan métodos para realizar el trabajo, Las prácticas de aseguramiento de calidad, como son muy robustas ya que aplican técnicas muy completas de medición, pueden proponer en un momento dado qué métodos se ajustan más a la naturaleza del producto a ser desarrollado, teniendo como efecto final que el producto tenga más posibilidades de ser un producto con calidad.

5. Se evita incurrir en costos innecesarios, Como un efecto generalizado de algunos de los puntos mencionados con anterioridad, la práctica de procesos de aseguramiento de calidad lleva a las organizaciones a evitar costos no deseados como pueden ser todos aquellos ocasionados por mantenimiento correctivo.

6. Se planea la calidad, Está claro que el concepto de calidad no es algo que se da de una manera automática e impredeciblemente. Es algo que se busca. Por lo mismo, se debe de planear, construir e implantar en el producto."[10]

2.7.COMPONENTES DE CALIDAD

"La Calidad es una meta específica que guía el proceso de desarrollo de un producto y/o servicio"[11]. Sin embargo, debe existir una medida para poder evaluar o determinar si un producto es de calidad o no.
Ishikawa menciona que para realizar la evaluación es importante contar con una manera de expresarla; o bien, un parámetro genérico para evaluarla y para establecer universalmente dicha parámetro.
Si no se tiene una medida estándar para evaluar no es posible medir, y menos comparar la calidad de dos diferentes productos.

Para resaltar la calidad es importante entender tres tipos de características que componen la expresión Calidad estas se detallan a continuación:

a) Características de calidad reales
b) Características de calidad implícitas
c) Características de calidad superiores

- Características de calidad reales

[10] [SIMMONS, RONALD A]
[11] [ISHIKAWA, KAORU]

Las características de calidad reales son aquellas que un usuario, espera de un producto y/o servicio.

- Características de calidad Implícitas

Las características implícitas Características implícitas son un conjunto de atributos que el usuario espera que su producto cumpla sin haber solicitado explícitamente el cumplimiento de estos atributos, es decir son características que se sobreentienden que un producto deba tener

- Características de calidad Superiores

Las *características superiores,* son todas aquellas que dan un valor agregado al producto solicitado por el usuario, estas características no son solicitadas por el usuario del producto pero brindan un valor agregado y dan mayor satisfacción al usuario.

Para sustentar lo mencionado veremos el siguiente ejemplo.
En un producto de software, las características de calidad reales se constituyen por una aplicación de software, libre de errores y que cumpla a cabalidad con los requerimientos del usuario.
Basados en el mismo ejemplo las características implícitas son: que la aplicación cuente con un análisis y un diseño que estén de acuerdo con las especificaciones del usuario, que la aplicación cuente con los principios de fiabilidad, seguridad, que sea fácil de mantener y que sea perdurable en el tiempo.
Finalmente las características superiores son: que brinde una buena experiencia de usuario, que no sea difícil de utilizar, que su interfaz sea amigable.

2.8.MÉTRICAS DE SOFTWARE.

Todas las entidades de software exitosas utilizan las mediciones como parte de sus actividades cotidianas, ya que estas brindan la información objetiva que permiten realizar una efectiva toma de decisiones y que tendrá un impacto positivo en el negocio y desempeño en la ingeniería.
Hay varias razones que justifican el uso de las métricas en el proceso de desarrollo de software. Por un lado se dice que cuando se puede medir aquello de lo cual se está hablando y se puede expresar en números, se sabe realmente acerca de ello; pero cuando no puede medirse, y no puede expresarse en números, el conocimiento que se tiene de ello es escaso e insatisfactorio.
Para poder asegurar que un proceso o sus productos resultantes son de calidad o poder compararlos, es necesario asignar valores, indicadores o algún otro mecanismo mediante el cual se pueda llevar a cabo dicha comparación.
Antes de entender el concepto de métrica es necesario aclarar que los términos, métricas, medición y medida no tienen el mismo significado.
A continuación presentamos una corta definición de cada uno de estos términos:

- Medida: Proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto.

- Medición: La medición es el acto de determinar una medida.
- Métrica: Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Las métricas nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto. *El objetivo de medir un producto es para intentar aumentar su calidad*.

La medición es muy común en el mundo de la ingeniería. Se mide, potencia de consumo, pesos, dimensiones físicas, temperaturas, voltajes, señales de ruidos por mencionar algunos aspectos. Desgraciadamente la medición se aleja de lo común en el mundo de la ingeniería del software. Encontramos dificultades en ponernos de acuerdo sobre que medir y como va evaluar las medidas.

Hay varias razones para medir un producto.

1. Para indicar la calidad del producto.
2. Para evaluar la productividad de la gente que desarrolla el producto.
3. Par evaluar los beneficios en términos de productividad y de calidad, derivados del uso de nuevos métodos y herramientas de la ingeniería de software.
4. Para establecer una línea de base para la estimación
5. Para ayudar a justificar el uso de nuevas herramientas o de formación adicional.

Las mediciones del mundo físico pueden englobarse en dos categorías: medidas directas y medidas indirectas.

Medidas Directas. En el proceso de ingeniería se encuentran el costo, y el esfuerzo aplicado, las líneas de código producidas, velocidad de ejecución, el tamaño de memoria y los defectos observados en un determinado periodo de tiempo.
Medidas Indirectas. Se encuentra la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento, etc.

2.9.CARACTERÍSTICAS DE LAS MÉTRICAS

Entre las características más importantes de las métricas del software es que debe ser objetiva, simple y calculable, consistente en el empleo de unidades y tamaños, persuasiva, además debería ser independiente del lenguaje de programación y proporcionar una realimentación eficaz para el desarrollador de software.

Las métricas deben ser un instrumento que ayude a mejorar el proceso, producto o proyecto de software, no tiene mucho sentido aplicar métricas que lejos de ayudar a los desarrolladores constituyan un problema; bien por ser demasiado complejas, porque no se entiendan correctamente los objetivos que persiguen o porque arrojen resultados imprecisos que no puedan ser interpretados por los ingenieros de software.

Lo más importante es que una métrica pueda obtenerse fácilmente, que se entienda por qué y para qué se utiliza, que los cálculos no produzcan resultados ambiguos o en los que existan extrañas combinaciones de unidades, finalmente que la interpretación de valores obtenidos esté acorde a las nociones intuitivas del ingeniero de software.

2.10. SIX SIGMA

2.10.1. DEFINICIÓN

Six Sigma es una metodología de la gerencia de calidad que provee a las empresas de herramientas para mejorar la calidad de sus procesos de negocio, este aumento en el desempeño y la disminución de la variación de los procesos conducen a la reducción de defectos y a la mejora de los beneficios, de la moral del empleado y de la calidad del producto.

Es una medición de la calidad y un programa de mejoramiento que fue iniciado por Mikel Harry y Motorola.

El Six Sigma se centra en el control de un proceso para llevarlo desde una línea de partida hasta el punto Six Sigma (desviaciones del estándar) es decir, a un nivel de 3.4 defectos por millón de productos producidos.

Incluye la identificación de los factores que son críticos para la calidad según lo determinado por el cliente. Reduce la variación del proceso y aumenta las capacidades de mejoramiento, de estabilidad y de diseño de sistemas para apoyar la meta del Six Sigma

El modelo Six Sigma es un acercamiento altamente disciplinario que puede ayudar a las compañías a centrarse en desarrollar y entregar productos y servicios casi perfectos.

Se basa en el trabajo estadístico, pionero en EEUU, de la gerencia de calidad la palabra Sigma es un signo griego usado como termino estadístico que indica hasta qué punto las medidas de un proceso en particular se desvían de la perfección mientras más alto sea el numero sigma más cerca se estará de la perfección.

2.10.2. HISTORIA

"Fue iniciado en Motorola el año 1982 por el ingeniero Bill Smith, como una estrategia de negocios y mejora de la calidad, pero posteriormente mejorado y popularizado por General Electric.

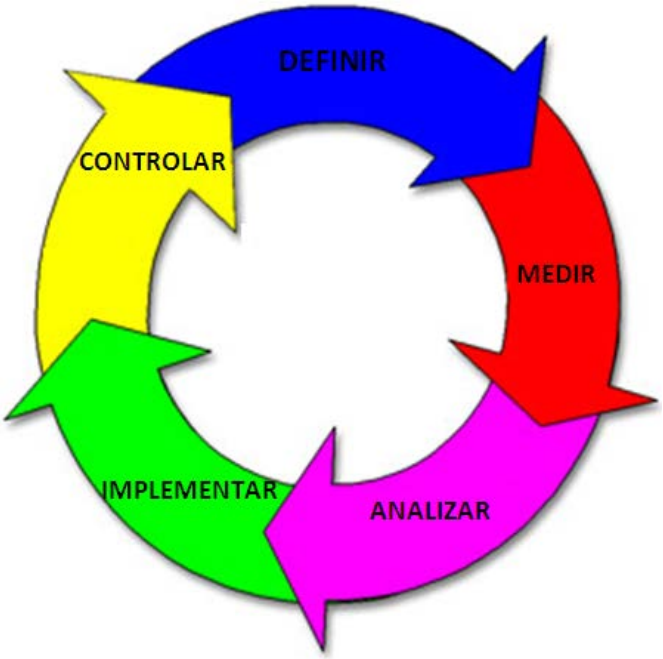
Los resultados para Motorola hoy en día son los siguientes: Incremento de la productividad de un 12,3 % anual; reducción de los costos de no calidad por encima de un 84,0 %; eliminación del 99,7 % de los defectos en sus procesos; ahorros en costos de manufactura sobre los Once Billones de dólares y un crecimiento anual del 17,0 % compuesto sobre ganancias, ingresos y valor de sus acciones.

El costo en entrenamiento de una persona en Seis Sigma se compensa ampliamente con los beneficios obtenidos a futuro. Motorola asegura haber ahorrado 17 mil millones de dólares desde su implementación, por lo que muchas otras empresas han decidido adoptar este método."[12]

[12] [WIKIPEDIA Six Sigma]

2.10.3. PROCESO

El proceso **Seis Sigma** se caracteriza por 5 etapas:



1. **Definir** el problema o el defecto
2. **Medir** y recopilar datos
3. **Analizar** datos
4. **Mejorar**
5. **Controlar**

DEFINIR.- Se identifican los posibles proyectos Seis Sigma, que deben ser evaluados por la dirección para evitar la inadecuada utilización de recursos. Una vez seleccionado el proyecto, se prepara y se asignan la prioridad necesaria.

MEDIR.- La fase de medición consiste en identificar los requisitos claves de los clientes, las características clave del producto (o variables del resultado) y los parámetros (variables de entrada) que afectan al funcionamiento del proceso y a las características o variables clave. A partir de esta caracterización se define el sistema de medida y se mide la capacidad del proceso.

ANALIZAR.-En esta fase, el equipo evalúa los datos de resultados actuales e históricos. Se desarrollan y comprueban hipótesis sobre posibles relaciones causa-efecto utilizando las herramientas estadísticas pertinentes. De esta forma el equipo confirma los determinantes del proceso, es decir las variables clave de entrada o "pocos vitales" que afectan a las variables de respuesta del proceso.

MEJORAR.-En la fase de mejora el equipo trata de determinar la relación causa-efecto para predecir, mejorar y optimizar el funcionamiento del proceso. Por último se determina el rango operacional de los parámetros o variables de entrada del proceso.

CONTROLAR.-Fase, control, consiste en diseñar y documentar los controles necesarios para asegurar que lo conseguido mediante el proyecto Seis Sigma se

mantenga una vez que se hayan implementado los cambios. Cuando se han logrado los objetivos y la misión se dé por finalizada, el equipo informa a la dirección y se disuelve.

RESULTADOS.-Conceptualmente los resultados de los proyectos Seis Sigma se obtienen por dos caminos. Los proyectos consiguen, por un lado, mejorar las características del producto o servicio, permitiendo conseguir mayores ingresos y, por otro, el ahorro de costos que se deriva de la disminución de fallas o errores y de los menores tiempos de ciclo en los procesos.

CAPÍTULO 3.- ACTIVIDADES DE CONTROL DE CALIDAD DEL SOFTWARE

3.1. ASPECTOS IMPORTANTES DEL CONTROL DE CALIDAD DEL SOFTWARE

El objetivo de las actividades de Control de Calidad es comprobar si un producto posee o no posee una determinada característica de calidad.
Cuando un producto no cumple con esta característica se dice que tiene un DEFECTO.
Por lo tanto, se puede decir que el objetivo del Control de Calidad del software es identificar defectos en el producto y corregirlos.

Hasta la fecha no se ha desarrollado ninguna teoría universalmente aceptada acerca de la prueba de software. Lo único que hay hasta el día de hoy es un conjunto de aproximaciones metodológicas que facilitan y hacen más eficiente el proceso de pruebas.

Se llama PRUEBA del Software al proceso en el que se ejecuta un sistema con el objetivo de detectar fallos.
Probar bien un sistema no es una actividad compleja de realizar. Algunas personas lo consideran un arte y aprender a hacerlo bien, requiere práctica y experiencia. Es importante considerar que el 50% del tiempo y esfuerzo del desarrollo de Software corresponde al tiempo de ejecución de pruebas para sistemas considerados como críticos.

Se llama DEPURACIÓN al proceso en el que se localiza el defecto que es la causa de un fallo, y se realiza una corrección. Por lo general, después del proceso de depuración es necesario repetir el proceso de prueba, para garantizar que el defecto quedó corregido y la corrección no provoco nuevos defectos.

"En un proyecto grande la prueba se puede llevar hasta el 50 o 60% del esfuerzo dedicado al proyecto."[13]
Por eso es muy importante seleccionar bien las pruebas que se van a realizar, teniendo en cuenta que las pruebas realmente valiosas son aquellas que localizan defectos.

Es importante tener en cuenta que el objetivo del proceso de prueba no es demostrar que el software está libre de defectos, sino precisamente descubrir defectos.
Por esta razón es necesario seleccionar casos de prueba que sean efectivos en la detección de defectos, mientras más defectos se encuentren en la ejecución de un caso de prueba se puede decir que la prueba es más exitosa.

[13] [ITEC]

Existen 3 objetivos fundamentales dentro de las actividades de control de calidad estos son los siguientes:

Objetivo a corto plazo.

- Creación de un equipo independiente de pruebas
- Definición de estrategias de pruebas
- Inicio de controles dinámicos de control (ejecución de pruebas de software sobre la aplicación)

Objetivos a mediano plazo.

- Gestión de pruebas con herramientas
- Definición de métricas de pruebas
- Lanzamiento de pruebas estáticas
  - Dentro de esta categoría tenemos el análisis estático automático y la verificación formal de programas.
  - La mayor parte del análisis estático automático del código lo realizan los compiladores, que pueden detectar desde expresiones sintácticamente incorrectas hasta incompatibilidades de tipo y otros errores de tipo semántico.
  - Otras técnicas de análisis estático automático de programas son:
    - Análisis de Flujo

      La técnica de análisis de flujo de datos fue introducida por Cocke y Allen, y consiste en trazar el comportamiento de las variables del programa desde su inicialización hasta que termina la ejecución del programa.
    - Ejecución simbólica

      Consiste en la ejecución simbólica de ciertos caminos dentro del programa, durante la cual se verifican ciertas expresiones simbólicas (expresiones compuestas por nombres de variables, en lugar de valores concretos para dichas variables), con respecto a ciertas condiciones y afirmaciones preestablecidas.

      El resultado de una ejecución simbólica es normalmente una expresión larga y compleja, pero la mejor forma de analizarla es descomponerla en una estructura de árbol, donde cada hoja representa un camino de ejecución y cada ramificación representa un punto de decisión en el programa.

      El árbol resultante se puede usar también como ayuda para generar casos de prueba.

Objetivos a largo plazo.

- Automatización del proceso de pruebas

3.2.TÉCNICAS DE PRUEBAS

3.2.1. Principios de las pruebas de software

Los principios de las pruebas con un conjunto de lineamientos de las pruebas de software, se ha realizado una recopilación de los lineamientos en los que se basan las pruebas de software

1. La prueba es el proceso de ejecutar un programa con la intención de encontrar errores (*una prueba exitosa es aquella que encuentra uno o más errores*).
2. Es imposible probar completamente cualquier módulo o cualquier sistema.
3. La prueba implica creatividad y trabajo duro.
4. La prueba puede prevenir posibles errores, cuando se realizan a través de las diferentes etapas del ciclo de vida.
5. Es mejor que las pruebas sean realizadas por personas diferentes a quienes hicieron el desarrollo del sistema.
6. Plantear y diseñar las pruebas antes de la generación de código.
7. El 80% de todos los errores se centran en solo en el 20% de la aplicación que se está probando
8. Una buena práctica es empezar las pruebas en módulos individuales y avanzar hasta probar el sistema entero.

3.2.2. Procedimientos generales para la ejecución de las pruebas de software

La persona encargada de realizar las pruebas (El tester) debe partir de estas premisas al momento de realizar las pruebas al software, se han detallado un conjunto de mejores prácticas que serán de utilidad para el encargado de las pruebas.

1. Usar siempre datos de entrada bien definidos al momento de comparar las salidas con los resultados esperados, cuando se ejecute una prueba.
2. Detectar primero los defectos obvios (usando datos de prueba muy simples) y luego sí realizar pruebas más complejas.
3. Cuando se modifique algo mientras se realiza la prueba es recomendable que se haga un solo cambio cada vez y que se utilicen los mismos datos con los que detectó el defecto, para la reprueba.
4. Probar el programa para verificar si se detectan entradas incorrectas y entradas calificadas como requeridas, es una muy buena práctica del probador de aplicaciones.
5. Seleccionar qué es lo que debe medir la prueba, es decir, cuál es su objetivo, para qué exactamente se hace la prueba.
6. Decidir cómo se va a realizar la prueba, es decir, qué clase de prueba se va a utilizar para medir la calidad escogida y qué clase de elementos de prueba se deben usar.
7. Desarrollar los casos de prueba. Un caso de prueba es un conjunto de datos o situaciones de prueba que se utilizarán para ejecutar una prueba o para revelar algo sobre el atributo de calidad que se está midiendo.
8. Determinar cuáles deberían ser los resultados esperados o correctos de los casos de prueba y crear un documento con los casos y sus resultados esperados,

9. Ejecutar los casos de prueba.
10. Documentar los resultados de las pruebas
11. Comparar los resultados de la prueba con los resultados esperados. Cualquier discrepancia entre ellos significa un error. Típicamente el error está en el sistema o unidad probada, pero también puede ser generado por algún aspecto del mismo proceso de prueba.

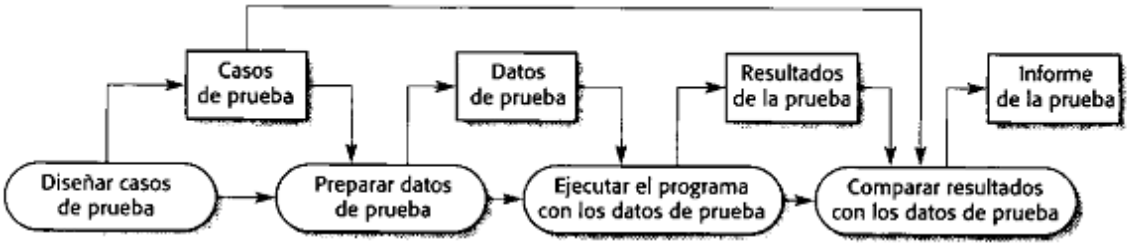


Figura 002.- Modelo del proceso de Pruebas[14]

3.2.3. Casos de Prueba

Un Caso de Prueba es una especificación, usualmente formal, de un conjunto de Entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de hacer una evaluación al elemento objeto de prueba.

Los Casos de Prueba buscan entre otras cosas evaluar los siguientes elementos:

- Los Caso de Prueba reflejan trazabilidad con los Casos de Uso (Funcionalidad), ya que estos muestran una secuencia ordenada de eventos, al describir flujos básicos, flujos alternos, precondiciones y postcondiciones.

- Las especificaciones suplementarias de requerimientos ya que existen otras características de calidad a evaluar, además de la funcionalidad, como Usabilidad, Confiabilidad, Eficiencia, Mantenibilidad y Portabilidad.

- Las especificaciones de diseño del Sistema, ya que se debe verificar que el software fue implementado según el diseño y que los elementos arquitectónicos garantizan la calidad del software.

Esto garantiza que los procedimientos de pruebas sean compatibles con las necesidades de los usuarios/clientes.

Cuando se sienta un tester a probar una aplicación, utilizando solo los casos de uso asume datos y procedimientos de pruebas sin que éstas queden documentadas.
Esto es un error ya que el Caso de Prueba extiende o amplía la información contenida en un Caso de Uso; por ejemplo, en el Caso de Uso no indicamos valores ni condiciones para la prueba.

[14] [SOMMERVILLE, Ian]

Los Casos de Prueba son esenciales para todas las actividades de pruebas por este motivo se clasifican 3 características muy importantes de los Casos de Prueba:

- Son la base para diseñar y ejecutar los procedimientos de pruebas.
- La profundidad de las pruebas es proporcional al número de casos de pruebas.
- El diseño y desarrollo, y los recursos necesarios son gobernados por los casos de pruebas requeridos.

Si los Casos de Prueba no son correctos, la Calidad del Sistema se pone en duda y las pruebas dejan de ser confiables.

El método de construir Casos de Prueba a partir de Caso de Uso parte del supuesto que se debe probar el comportamiento del software en base a las solicitudes o requerimientos.
Se ha propuesto en este trabajo un método para especificar Casos de Prueba a partir de Caso de Uso, el mismo que está constituido por cuatro fases principales:

1. Identificar Escenarios.
   Una vez entendidos y analizados los casos de Uso se identifican y se clasifican escenarios.

2. Identificar Casos de Prueba.
   Luego de identificados los escenarios se recurre a la parte imaginativa del probador ya que deberá imaginar al sistema bajo situaciones extremas, identificando así situaciones o casos en los que el sistema será probado

3. Especificar los Casos de Prueba.
   Se documentaran todos los escenarios imaginados en el punto dos, con información detallada como, datos de entrada, datos de salida, procesos que involucra, procesos alternos etc.…

4. Ejecución y Aprobación del Casos de Prueba.
   Finalmente y de una manera coordinada y sistematizada se procede a ejecutar los casos de prueba.

El presente Trabajo propone una plantilla sobre los cuales se deberá documentar los Casos de Prueba… ***Referirse al Anexo 001***

3.2.4. Prueba de Caja Negra

Las pruebas de caja negra tienen por objetivo probar como si la aplicación fuera una caja negra, quiere decir que los casos de prueba se basarán en el conocimiento acerca de la funcionalidad deseada (descripción funcional), para esta prueba los procesos internos no son tan importantes como la salida del proceso, el mismo que será validado con el resultado esperado.
A la mayor parte de los usuarios de programas extensos no les interesa los detalles de su funcionamiento; lo único que desean es conseguir respuestas. Es decir, desean tratar el programa como una caja negra a la cual le introducen datos de entrada y obtienen de ella

los datos de salida que esperan. De ahí el nombre de este método. De manera análoga, los datos de prueba se escogerán atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que éste corra bien.

A la prueba de caja negra también se le llama prueba funcional o prueba orientada al diseño.

Una prueba de caja negra exhaustiva requiere de la generación de un caso de prueba por cada combinación válida o no válida de los valores de entrada, lo cual resulta imposible en la mayor parte de los casos por la cantidad de combinaciones resultantes.
Por eso se utilizan diferentes criterios a la hora de restringir el conjunto de casos de prueba.

Para la selección del conjunto de casos de prueba más usuales se tomara en cuenta los siguientes métodos:

- *Método de clases de equivalencia*
  Consiste en dividir las posibles entradas al sistema en clases equivalentes de tal forma que bastara con probar una sola vez los elementos la clase equivalente.

- *Grafos causa/efecto y tablas de decisión*
  Consiste en crear un grafo causa/efecto a partir de las especificaciones, y seleccionar suficientes casos de prueba como para asegurar la cobertura del grafo. Se llama causas a las características de los datos de entrada y efectos a las clases de salidas que puede proporcionar el programa.
  A partir del grafo causa/efecto se construye una tabla de decisión que refleje las dependencias entre causas y efectos. Lo que se hace entonces es reducir la tabla de decisión y seleccionar sólo un caso de prueba para todas las causas que producen el mismo efecto, o para cada columna de la tabla de decisión.

- *Adivinación de errores*
  Consiste en tratar de imaginar cuáles son los errores que se pueden haber cometido con mayor probabilidad, y generar casos de prueba para comprobar dichos errores.

A continuación se detallara algunas características del método de Caja Negra

- Forman parte de las pruebas directas
- Consisten en la elaboración de un conjunto de casos de prueba
- La salida esperada forma parte del caso de prueba
- Los casos de prueba deben contener tanto datos válidos como datos no válidos para el programa
- La única información disponible para su elaboración es la especificación original del programa
- La falta de acceso al código permite detectar errores en la interpretación de la especificación

Cuáles son los criterios mínimos que el especialista de pruebas debe tener en cuenta al elegir los datos de prueba para los programas:

1. **Valores fáciles** El programa se depurará con datos que sean fáciles de comprobar.
2. **Valores típicos realistas** Siempre se ensayará un programa con datos seleccionados para que representen cómo se aplicará. Tales datos han de ser suficientemente sencillos, de modo que los resultados sean verificables en forma manual.
3. **Valores extremos** Muchos programas cometen errores en los límites de sus rangos de aplicaciones. Es muy fácil que las instrucciones que incluyen a los contadores de ciclos o que hacen referencian a las dimensiones de un arreglo se equivoquen por uno.
4. **Valores ilegales** "Basura entra, basura sale" es un viejo refrán en los círculos de computación y conviene respetarlo. Cuando en un programa entra basura, su reacción inmediata habrá de ser por lo menos un mensaje de error adecuado para el usuario.

El Método de la Caja Negra está orientado a encontrar los siguientes defectos:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

3.2.5. Prueba de Caja Blanca

"La prueba de caja blanca, denominada a veces *prueba de caja de cristal* es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

1. Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo;
2. Ejerciten todas las decisiones lógicas en sus vertientes verdadera **y** falsa;
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales;
4. Ejerciten las estructuras internas de datos para asegurar su validez."…[15]

Las pruebas de caja blanca son muy importantes partiendo de la siguiente premisa "un programa difícilmente puede considerarse probado por completo, si su código contiene partes que nunca han sido ejecutadas."

En las pruebas de Caja blanca, se analiza la estructura lógica del programa. Así pues, se procura escoger los datos que verifiquen cada posibilidad en las estructuras de selección múltiple, como las cláusulas de cada proposición **if**, cada alternativa de cada instrucción **case (switch)** y la condición de terminación de cada ciclo.

Cuando se prueba un programa extenso, este método es sin duda impráctico, pero en un módulo pequeño constituye un excelente método de pruebas y depuración.

[15] [PRESSMAN. R]

En un programa bien diseñado, cada módulo incluirá unos pocos ciclos y opciones. De ahí que son suficientes algunos casos de prueba bien seleccionados para probar cada módulo por separado.

En este tipo de métodos, el objeto que se desea probar se ve como una caja blanca. Esto quiere decir que la elección de los casos de prueba se va a basar en el conocimiento que se tenga acerca de la estructura del objeto (diseño detallado, diagramas de flujo de datos y de control, código fuente).
A la prueba de caja blanca también se le llama prueba estructural, Se utilizan para complementar los resultados de las técnicas de caja negra

Para que una prueba de caja blanca sea considerada como exhaustiva requerirá la generación de un caso de prueba por cada posible camino. Como esto no es posible, por lo general, se utilizan métricas que dan una indicación de la calidad de un determinado conjunto de casos de prueba. Las métricas más utilizadas son:

- Cobertura de sentencias
- Cobertura de segmentos entre decisiones.
- Cobertura de decisiones de ramificación
- Cobertura de condiciones
- Cobertura de todas las combinaciones de condiciones
- Cobertura de caminos

3.3.ESTRATEGIAS DE PRUEBAS

3.3.1. Verificación y Validación

"Probar un sistema es la forma más común de comprobar que un sistema satisface su especificación y hace lo que el cliente quiere que haga, sin embargo las pruebas son solo una de las técnicas de verificación y validación.
Algunas de estas técnicas no se han convertido en tendencias principales de la ingeniera de software"[16]

La Validación o la verificación no son lo mismo, aunque a menudo se confunden estos dos términos.

Que es verificación?

La verificación se refiere a la construcción correcta. Se puede definir verificación como el proceso de determinar si la lógica operacional del modelo (programa de ordenador) se corresponde con la lógica del diseño. En términos más simples, consiste en determinar si hay errores en el programa. En otros términos y como dice Sommerville la verificación debe responder a la pregunta "Estamos construyendo el producto correctamente" [16]
Esta definición nos dice claramente que el objetivo de la verificación implica comprobar que el SOFTWARE cumple a cabalidad con su especificación, deberá comprobarse si el sistema funciona o no funciona.

[16] [SOMMERVILLE, Ian]

Que es la validación?

La validación es el proceso de determinar si el sistema, cumple con lo solicitado por el cliente.
Usualmente la validación se consigue a través del afinamiento de las solicitudes y requerimientos del cliente.
Tal como lo hizo con la verificación recuerdo nuevamente a Sommerville cuando en su libro dice que la validación deberá responder a la pregunta "Estamos construyendo el producto correcto" [16]

La validación es un proceso muy generalizado, se trata de asegurar, comprobar que el sistema cumple con las expectativas del cliente, es decir comprobar que el sistema hace lo que el cliente espera que haga su sistema

Podemos resumir el papel que la verificación y la validación desempeñan en la etapa de diseño del software, en el siguiente cuadro.

Modelo	Verificación	Validación
Modelo Conceptual		- Contiene el modelo todos los elementos, sucesos y relaciones relevantes - Podrá el modelo responder a las cuestiones planteadas
Modelo Lógico	- Están los eventos representados correctamente - Son las formulas matemáticas y las relaciones correctas - Están las medidas estadísticas formuladas correctamente	- Incluye el modelo todos los elementos considerados en el modelo conceptual - Contiene todas las relaciones del modelo conceptual
Modelo de Simulación	- Contiene el código todos los aspectos del modelo lógico - Están las estadísticas y las formulas calculadas correctamente - Contiene el modelo errores de codificación	- Es el modelo una representación valida del sistema real - Puede el modelo duplicar el comportamiento del sistema real - Es creíble el modelo para los expertos del sistema

Figura 003.- Cuadro Comparativo de Verificación y Validación

Dentro del proceso de validación y verificación existen dos temas complementarios para el análisis y comprobación del sistema.
Así tenemos:

- Las Inspecciones de software, "Comprueban las representaciones del sistema, como el documento de requerimientos, los diagramas de diseño y el código fuente del sistema"[17]

[17] [SOMMERVILLE, Ian]

Las inspecciones de software están diseñadas para usarse en cualquier etapa del proceso de desarrollo, así como también muestran la flexibilidad de complementarse con algún tipo de análisis automático del código fuente de un sistema o de los documentos asociados.
Las inspecciones pueden ser consideradas como una técnica estática ya que no se necesita de la ejecución del software

- Las pruebas de software, implica ejecutar una instancia del software con datos de prueba, examinando su comportamiento y sus salidas
  Estas pruebas de software son consideradas como una técnica dinámica de verificación y validación

  La figura a continuación, tomada del libro de Somerville da la premisa para pensar que las inspecciones de software y las pruebas de software son actividades complementarias.

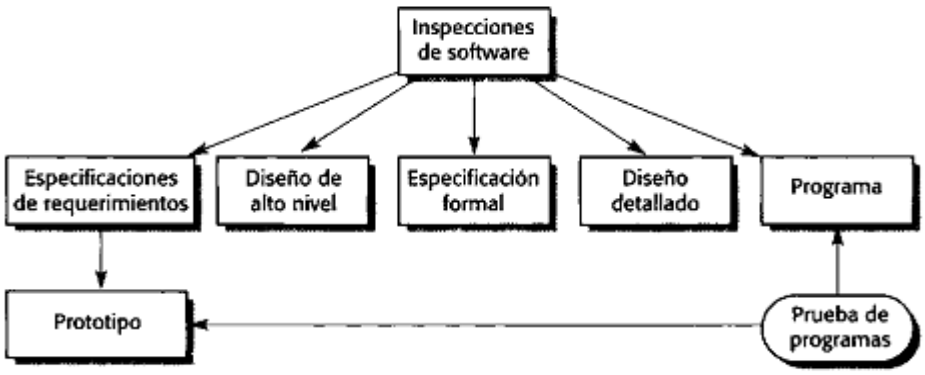


Figura 004.- Verificación y validación estática y dinámica[17]

3.3.2. Pruebas del Software

"Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón, se debe definir en el proceso de la ingeniería del software una plantilla para las pruebas del software: un conjunto de pasos en los que podamos situar los métodos específicos de diseño de casos de prueba."…[18]

Sommerville menciona en su libro que "Existen dos tipos distintos de pruebas, los mismos que pueden utilizarse en diferentes etapas del proceso de desarrollo del software" y las clasifica de la siguiente manera.

- Pruebas de Validación su objetivo es demostrar que el software desarrollado es el que el cliente quiere, que satisface sus requerimientos.

[18] [PRESSMAN. R]

- Pruebas de defectos su objetivo es revelar inconsistencias entre el programa y su especificación.

Afortunadamente esta clasificación no es excluyente, ya que con el objetivo de garantizar la calidad del sistema se puede ejecutar los dos tipos de pruebas y estas pueden encontrar inconsistencias, así como garantizar que los requerimientos del cliente se satisfagan correctamente.

3.3.2.1. Tipos De Prueba

De acuerdo con el estándar IEEE 1012-1986 el conjunto mínimo de pruebas que se deben realizar son:

- **Prueba modular, prueba unitaria o prueba de componentes**: Consiste en la prueba que se debe realizar a cada módulo, aislado del resto del sistema.

- **Prueba de integración:** se realiza a medida que los diferentes módulos del sistema se integran en el mismo. Luego de las pruebas modulares se supone que todos módulos están correctos.
  El objetivo fundamental de esta prueba es comprobar que las interfaces entre los distintos módulos son correctas.

- **Prueba del sistema:** se realiza cuando se han integrado todos los módulos, y su objetivo es comprobar que el sistema satisface los requisitos del usuario, tanto los funcionales como los no funcionales.

- **Prueba de aceptación:** Se realiza una vez que el sistema se ha implantado en su entorno real de funcionamiento, y su objetivo es demostrar al usuario que el sistema satisface sus necesidades.

- **Pruebas de regresión:** El objetivo es comprobar que toda nueva versión de un producto software es de no menos calidad que la versión anterior, es decir, que al introducir cambios no se ha reducido la valoración de ninguna de las características de calidad que tenía el producto.

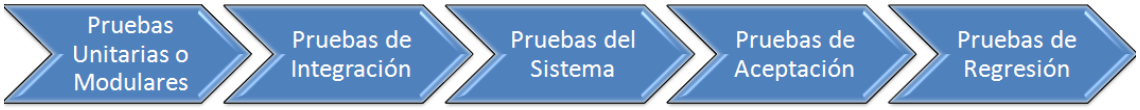


Figura 005.- Secuencia de la ejecución de las Pruebas de Software

3.4. METODOLOGÍA DE TESTING

Cada uno de los diferentes tipos de prueba implica la realización de un conjunto de actividades estándar, así como la producción de un conjunto de salidas estándar.

3.4.1. Actividades estándar de las pruebas de software

Las actividades estándar de pruebas son un conjunto de pasos que generalmente se realizan al momento de probar una aplicación. De esta forma se garantiza que se sigan un conjunto de actividades consecutivas y establecidas al momento de realizar una prueba.
Las actividades son:

1. Planificación de la prueba.
2. Diseño de la prueba.
3. Determinación de los casos de prueba.
4. Planificación del procedimiento de prueba
5. Ejecución de la prueba
6. Análisis y evaluación de la prueba.

A continuación veremos cada una de estas actividades en detalle.

1. Planificación de la prueba: Esta actividad consiste en la creación de un plan de pruebas en el que se registra:
    - El objetivo del proceso de prueba.
    - Los objetos que hay que probar.
    - Las características que se van a probar y las que no.
    - El método de prueba a utilizar.
    - Los recursos que se van a emplear.
    - El plan de tiempos.
    - Los productos a generar durante las pruebas
    - El reparto de las responsabilidades.

2. Diseño de la prueba: Esta actividad consiste en dar instrucciones detalladas acerca de:
    - Cómo llevar a cabo la prueba para alcanzar los objetivos deseados,
    - De qué forma se van a utilizar los métodos de prueba,
    - Qué objetos se van a probar en cada una de las pruebas y
    - Qué criterios se van a utilizar para determinar si el objeto pasa o no pasa la prueba.

3. Determinación de los casos de prueba: Esta actividad consiste en especificar el conjunto de casos de prueba a utilizar en función del diseño realizado para la prueba. Para cada caso de prueba habrá que especificar:
    - Qué objetos se van a probar,
    - Qué entradas se les van a dar y
    - Cuáles son las salidas esperadas.

4. Planificación del procedimiento de prueba: Esta actividad consiste en fijar un conjunto de pasos para la ejecución de la prueba. Se especifica detalladamente:

- La secuencia exacta de ejecución de los distintos casos de prueba,
- Los requisitos que hay que cumplir para la ejecución de cada caso y
- Las condiciones de terminación de cada uno de ellos.

5. Ejecución de la prueba: Esta actividad consiste en ejecutar cada caso de prueba, según el procedimiento especificado en el paso anterior, y registrar los incidentes o problemas encontrados durante la misma.

6. Análisis y evaluación de la prueba: Se examinan los resultados de la prueba y se decide si se han alcanzado los objetivos propuestos o si se debe repetir la prueba.

3.5. HERRAMIENTAS PARA AUTOMATIZAR LAS PRUEBAS DE SOFTWARE

Son aplicaciones que existen en el mercado de la tecnología y que por medio de su utilización se puede optimizar tiempo y recursos, además el uso de estas herramientas permite:

- Gestionar el proceso de testeo,
- Generar datos de prueba,
- Automatizar las pruebas
- Evaluar resultados,
- Grabar escenarios dirigidos por el “tester”
- Permiten la regresión de casos de Prueba
- Permite optimizar tiempo
- Enfocar el esfuerzo de las pruebas de software a sectores críticos de la aplicación

3.5.1. Visual Studio Team Test

3.5.1.1. Características

- Permiten grabar una interacción
- Permiten establecer parámetros cambiantes
- Almacenamiento de documentos del proyecto
- Validación de requisitos no funcionales
- Validación de la arquitectura
- Por medio de la ejecución de las pruebas de carga y rendimiento permiten asegurar la escalabilidad a futuro
- Registro de Fallas –Bugs y administración de defectos

3.5.1.2. Tipos de Prueba que permite ejecutar

- Prueba Unitaria
- Prueba web
- Prueba de carga
- Perfilamiento de código

3.5.2. IBM- Rational Robot

3.5.2.1. Características

- Facilita la transición de los equipos de pruebas manuales a pruebas automatizadas. La realización de pruebas de regresión con IBM Rational Robot es un paso adelante hacia la automatización, ya que resulta fácil de utilizar y ayuda a los equipos de pruebas a aprender los procesos de automatización mientras trabajan.
- Permite a los ingenieros expertos en automatización de pruebas identificar más defectos al ampliar sus scripts de pruebas con lógica condicional, para abarcar una mayor parte de la aplicación y para definir casos de prueba con llamadas a bibliotecas DLL o ejecutables externos.
- Incluye un componente de gestión de pruebas y se integra con las herramientas de IBM Rational Team Unifying Platform para realizar el seguimiento de defectos, gestión de cambios y rastreo de requisitos.
- Simplifica la configuración de pruebas. IBM Rational Robot puede ser utilizado para realizar pruebas distribuidas entre muchas máquinas, cada una configurada diferente. Las mismas pruebas funcionales pueden ser ejecutadas simultáneamente, acortando el tiempo para identificar problemas con configuraciones específicas.
- Prueba diferentes tipos de aplicaciones. IBM Rational Robot soporta un amplio rango de ambientes y lenguajes, incluyendo HTML y DHTML, Java™, VS.NET, Microsoft Visual Basic y Visual C++, Oracle Developer/2000, PeopleSoft, Sybase PowerBuilder y Borland Delphi.
- Pruebas de controles y objetos. permite probar cada componente de la aplicación bajo una variedad de condiciones y provee casos de prueba de menús, listas, caracteres, bitmaps.

3.5.2.2. Tipos de Prueba que permite ejecutar

- Pruebas Funcionales
- Automatiza pruebas funcionales
- Automatiza pruebas de regresión para aplicaciones e-commerce, cliente/servidor y ERP.
- Pruebas de Regresión
- Aplicaciones Web
- Aplicaciones Windows

3.5.3. QAwizard Pro

3.5.3.1. Características

- Permite la automatización de pruebas funcionales y de regresión
- Funciona tanto para aplicaciones web como para aplicaciones Windows

- Es muy útil ya que proporciona reportes luego de cada prueba permitiendo al tester mantener una adecuada documentación sobre las pruebas realizadas.

3.5.3.2. Tipos de Prueba que permite ejecutar

- Pruebas Funcionales
- Pruebas de Regresión
- Aplicaciones Web
- Aplicaciones Windows

3.5.4. Ants Profiler

3.5.4.1. Características

- Permite realizar un perfilamiento del código que se utiliza en las aplicaciones escritas en cualquier lenguaje de .Net
- Es muy optimo para optimizar el rendimiento de la aplicación que se está probando
- Permite identificar segmentos de código que no están trabajando
- Permite verificar el uso de la memoria
- Permite verificar las clases y los objetos que utilizan más recursos
- Permite identificar el objeto que es más utilizado dentro de la aplicación
- Es muy útil para las aplicaciones web y Windows desarrolladas con . Net
- Permite identificar el método más utilizado
- Permite optimizar el rendimiento de la aplicación

3.5.4.2. Tipos de Prueba que permite ejecutar

- Aplicaciones de 32-bit o 64-bit
- Aplicaciones Web
- Aplicaciones Windows

CAPITULO 4.- GESTIÓN DE PROYECTOS

Las actividades de gestión de proyectos de Software son una parte esencial de la ingeniería de software, generalmente una buena gestión no asegura el éxito ni la calidad de un proyecto, sin embargo una mala gestión usualmente lleva al fracaso del proyecto, pero qué tanto influye la gestión de un proyecto en la calidad del mismo? Influye de tal manera que, cuando un proyecto cuenta con una gestión adecuada, se realiza con una metodología, una planificación previamente establecida.

4.1. ACTIVIDADES DE GESTIÓN DE UN PROYECTO

Resulta realmente imposible describir exactamente el trabajo de un gestor de proyectos de Software, sin embargo existe un conjunto de actividades comunes detalladas en los escritos de Ian Somerville. Las mismas que revisaremos a continuación.

- Redacción de la propuesta
- Planificación y calendarización del proyecto
- Estimación de costes del proyecto
- Selección y evaluación del personal
- Redacción y presentación de informes

4.1.1. Redacción de la Propuesta

Esta primera etapa implica redactar un escrito, el mismo que servirá para proponer la idea del proyecto al cliente con el que se esté trabajando, la propuesta describe los objetivos del proyecto y la forma en que se va a realizar el trabajo para cumplir con los objetivos y requerimientos del cliente.

La redacción de una propuesta es una tarea critica ya que este es el punto de partida y el generador de trabajo en la mayoría de empresas que desarrollan software, de una propuesta de proyecto bien realizada y bien redactada dependerá que los contratos sean asignados y aprobados a la empresa que prestaría el servicio.

No existe una guía o un libro donde se detalle de qué manera se deberá redactar una propuesta, el éxito de este trabajo está basado en la práctica, y experiencia, del gerente de proyectos.

4.1.2. Planificación y Calendarización del proyecto

Se refiere a la identificación de actividades, hitos y entregas de un proyecto, de esta manera el gerente de proyecto deberá proyectar un plan para guiar el desarrollo hacia las metas del proyecto previamente establecidas en la propuesta.

4.1.3. Estimación de Costes del Proyecto

La estimación de costos, es una actividad ligada con la gestión de personal que estará involucrado en el proyecto, este aspecto se tratara más adelante.

La estimación es más arte que ciencia; también es parte de la etapa de la planificación y algunas actividades de la ingeniería. La diferencia en la estimación de costos entre ingeniería de software y otras disciplinas es que en ingeniería de software lo principal para las personas es el costo; y en otras disciplinas el costo de las cosas materiales depende de la actividad.

Existen técnicas para la estimación de costos, pero para ello se requiere experiencia, acceso a una buena información histórica y coraje para confiar en medidas cuantitativas cuando todo lo que existe son datos cualitativos.
El costo principal para un proyecto de desarrollo de software es sin duda el tamaño del producto. La medida del tamaño debe ser tal que esté en relación directa con el esfuerzo de desarrollo, por lo que las métricas de tamaño tratan de considerar todos los aspectos que influyen en el costo, como tecnología, tipos de recursos y complejidad.

4.1.4. Selección y evaluación del personal

Esta actividad dependerá del tipo de talento humano con el que cuenta la empresa, antes de continuar con el proyecto el gerente de proyecto deberá estar en la capacidad de evaluar si el personal de su empresa está apto para cumplir con las tareas planificadas que permitirán cumplir con el objetivo del proyecto.

4.1.5. Redacción y presentación de Informes

Así como el gerente del proyecto es el encargado de presentar una propuesta al cliente, deberá también estar en la capacidad de informar a sus superiores sobre el estado de los proyectos, para esto es muy importante que el gerente conozca de buena forma el estado del proyecto, mediante revisiones formales, informales, entrevistas con el personal involucrado. Etc.…

4.2.PLANIFICACIÓN DEL PROYECTO

La gestión efectiva de un proyecto de software depende de planificar completa y correctamente el progreso del proyecto, un plan bien elaborado y preparado al inicio de un proyecto deberá utilizarse como el camino por el cual se va a conducir el proyecto este plan inicial deberá ser l mejor posible de acuerdo con la información disponible el mismo que mejorara y evolucionara conforme el progreso del proyecto.

4.2.1. PLAN DEL PROYECTO

El proceso de la planificación del proyecto inicia con una evaluación de las restricciones que afectan al proyecto (fechas entrega, personal disponible, presupuesto asignado, etc.).
Luego de analizadas las restricciones se procederá entonces a definir un conjunto de hitos y entregables.
El plan de proyecto fija los recursos disponibles, divide el trabajo y crea un calendario de trabajo.

4.2.2. HITOS Y ENTREGAS

Los gerentes de proyectos para realizar su trabajo necesitan de información, el Software es intangible, y la información que sirve a los gerentes de proyectos solo se puede proveer en forma de documentos o entregables
Cuando se planifica un proyecto se deben establecer una serie de hitos, que no son nada más sino el punto límite entre una actividad y otra, indicando así que el tiempo de ejecución de una tarea ha finalizado, dejando como evidencia de esta finalización un producto entregable.
Los informes de los hitos no deben ser documentos amplios ni ambiguos sino más bien deben ser cortos u directos indicando los logros conseguidos en ese periodo de ejecución

4.3.CALENDARIZACIÓN DEL PROYECTO

La tarea principal del gerente del proyecto en esta actividad es el de estimar el tiempo y los recursos necesarios para completar las actividades y organizarlas en una sucesión coherente.
La experiencia del gerente de proyecto, y la de su equipo de trabajo sirven como un historial para las nuevas estimaciones que se realicen para los nuevos proyectos. Las tareas de estimación se pueden complicar ya que, por definición, ningún proyecto es igual a otro.
La calendarización del proyecto implica identificar las actividades, las dependencias de las actividades, estimar los recursos necesarios y las capacidades de los recursos para ejecutar las actividades, asignación de personal a las actividades.
Partiendo de este proceso se podrán realizar la calendarización de los proyectos.
Una buena práctica es estimar en el más ideal de los escenarios, es decir nada saldría mal luego de eso si se debe incrementar la estimación para abarcar los problemas previstos, partiendo de esta premisa también se debe agregar a la estimación un factor de contingencia adicional.
La mayoría de los proyectos de software fracasan por falta de tiempo de calendario.
Las causas de falta de tiempo pueden resumirse en las siguientes:

1. Las técnicas de estimación son pobremente desarrolladas. Reflejan suposiciones falsas, por ejemplo, que todo irá bien. Sin pensar en los planes de contingencia que se pueden aplicar.
2. Se confunde esfuerzo con progreso, suponiendo que hombres y meses son intercambiables.
3. El progreso de la Calendarización es pobremente monitoreado.
4. Cuando un resbalón en la Calendarización es reconocido, la respuesta tradicional es añadir mano de obra. Esto es similar a apagar un fuego con gasolina, ya que no siempre el agregar recursos es la solución a los problemas de calendarización.

4.4.GESTIÓN DE RIESGOS

"La **Gestión de riesgos** es un enfoque estructurado para manejar la incertidumbre relativa a una amenaza, a través de una secuencia de actividades humanas que incluyen evaluación de riesgo, estrategias de desarrollo para manejarlo y mitigación del riesgo utilizando recursos gerenciales. Las estrategias incluyen transferir el riesgo a otra parte, evadir el riesgo, reducir los efectos negativos del riesgo y aceptar algunas o todas las consecuencias de un riesgo particular."[19]

Esto se constituye en una tarea muy importante para el gerente del proyecto ya que debe identificar y anticiparse a los riesgos para que estos no afecten en la programación del proyecto o en la calidad del Software a desarrollar y aplicar algunas de las estrategias de la gestión de riesgo antes mencionadas.
Se define como riesgo como una probabilidad de que una circunstancia adversa ocurra.
Los riesgos se pueden clasificar de diferentes enfoques, a nivel del proyecto a nivel del desarrollo del Software que se está desarrollando, y para la organización como tal.

[19] [WIKIPEDIA Gestión de riesgos]

4.4.1. IDENTIFICACIÓN DE RIESGOS

Comprende en el descubrimiento de los posibles riesgos del proyecto esta etapa solo se centra en identificar los riesgos y clasificarlos, inicialmente no se tomaran en cuenta los riesgos de baja probabilidad. Como ayuda a la identificación de los riesgos en el libro de Ian Somerville se presenta un cuadro con los posibles riesgos del Software.

Rotación de personal	Proyecto	Personal con experiencia abandona el proyecto antes de que finalice.
Cambio de gestión	Proyecto	Habrá un cambio de gestión organizacional con diferentes prioridades.
No disponibilidad del hardware	Proyecto	El hardware esencial para el proyecto no será entregado a tiempo.
Cambio de requerimientos	Proyecto y producto	Habrá más cambios en los requerimientos de lo esperado.
Retrasos en la especificación	Proyecto y producto	Las especificaciones de las interfaces esenciales no estarán a tiempo.
Subestimación del tamaño	Proyecto y producto	El tamaño del sistema se ha subestimado.
Bajo rendimiento de la herramienta CASE	Producto	Las herramientas CASE que ayudan al proyecto no tienen el rendimiento esperado.
Cambio de tecnología	Negocio	Un producto competitivo se pone en venta antes de que el sistema se complete.
Competencia del producto	Negocio	La tecnología fundamental sobre la que se construirá el sistema se sustituye por nueva tecnología.

Posibles riesgos del software.

Figura 006.- Cuadro de posibles riesgos del software[20]

La Identificación de Riesgos es un proceso iterativo ya que se pueden descubrir nuevos riesgos a medida que el proyecto avanza a lo largo de su ciclo de vida del desarrollo del proyecto.

La frecuencia de la iteración y quién participará en cada ciclo variará dependiendo de la importancia del proyecto y de las definiciones iníciales que se hayan realizado al inicio del mismo.

El equipo del proyecto debe participar en el proceso para poder desarrollar y mantener un sentido de pertenencia y responsabilidad por los riesgos y la respectiva gestión que se dará a los mismos.

Las personas externas al proyecto pueden proporcionar información extra sobre los objetivos. El proceso Identificación de Riesgos dispara automáticamente el procesos sobre el Análisis Cualitativo de Riesgos (El Análisis Cualitativo de Riesgos es una forma rápida y rentable de establecer prioridades para la Planificación de la Respuesta a los Riesgos). Adicional a este primer proceso de análisis y como alternativa, se tiene el proceso de Análisis Cuantitativo de Riesgos (analiza el efecto de esos riesgos y les asigna una calificación numérica.) cuando lo dirige un director de riesgos experimentado.

En algunas ocasiones, simplemente la identificación de un riesgo puede sugerir su respuesta, y esto debe registrarse para realizar otros análisis.

[20] [SOMMERVILLE, Ian]

4.4.2. ANÁLISIS DE RIESGOS

Durante la etapa de análisis de riesgos se considera por separado cada uno de los riesgos identificados y se evalúa acerca de la probabilidad y la seriedad del mismo, la única forma de realizar correctamente esta actividad es basándose en la experiencia del gerente de proyecto, la valoración no se realiza con una numeración precisa sino con unos porcentajes de probabilidad de ocurrencia la probabilidad del riesgo se puede valorar como muy bajo <10%, bajo (10-25%), moderado (25-50 %), alto (50-75%), o muy alto (>75%).
Los efectos del riesgo se pueden categorizar de la siguiente forma:

- Catastrófico
- Serio
- Tolerable
- Insignificante

El resultado del análisis del riesgo se verá reflejado en una tabla la cual, debe estar ordenada según la seriedad del riesgo, cada riesgo deberá también estar acompañado de una estrategia de prevención, minimización y de contingencia.
Finalizado el análisis de los riesgos del proyecto es necesario realizar una clasificación de los riesgos considerados los más críticos e importantes, basados en la probabilidad y el efecto de los riesgos identificados.

4.4.3. PLANIFICACIÓN DE RIESGOS

El proceso de Planificación de riesgos considera cada uno de los riesgos claves o críticos que han sido identificados, así como la estrategia para gestionarlos.
Este proceso no tiene una metodología y mucho menos una guía, sencillamente depende de la experiencia y habilidad del gerente del proyecto.
"Estrategias para los riesgos identificados.

- Estrategia de Prevención El objetivo es reducir la probabilidad de que el riesgo aparezca.
- Estrategias de Minimización el objetivo de esta estrategia es reducir el impacto del riesgo
- Planes de Contingencia El objetivo es estar preparado para el peor de los escenarios una vez de que el riesgo se concrete.

La Planificación del Riesgos aborda los riesgos en función de su prioridad, introduciendo recursos y actividades en el presupuesto, cronograma y plan de gestión del proyecto, según sea necesario."[21]

Las respuestas a los riesgos planificadas deben ser congruentes con la importancia del riesgo, tener un coste efectivo en relación al desafío, ser aplicadas a su debido tiempo, ser realistas dentro del contexto del proyecto, estar acordadas por todas las partes implicadas, y

[21] [SOMMERVILLE, Ian]

a cargo de una persona responsable. A menudo, es necesario seleccionar la mejor respuesta a los riesgos entre varias opciones.
La Planificación de la Respuesta a los Riesgos presenta los enfoques comúnmente usados para planificar las respuestas a los riesgos. Los riesgos incluyen las amenazas y las oportunidades que pueden afectar al éxito del proyecto, y se discuten las respuestas para cada una de ellas.

4.4.4. SUPERVISIÓN DE RIESGOS

La supervisión de los riesgos se realiza a medida que avanza el proyecto en esta actividad de la gestión de riesgos se evalúa todos los riesgos inscritos en la matriz de riesgos y se verifica que su probabilidad de ocurrencia no haya cambiado.
Como se ha mencionado la supervisión de los riesgos debe ser un proceso continuo y se lo debe revisar en cada una de las reuniones de seguimiento planificadas.
Importante aclarar que cada uno de los riesgos debe ser considerado y analizado por separado

La Supervisión del riesgo es el proceso de identificar, analizar y planificar nuevos riesgos, realizar el seguimiento de los riesgos identificados y los que se encuentran en la lista de supervisión, volver a analizar los riesgos existentes, realizar el seguimiento de las condiciones que disparan los planes de contingencia, realizar el seguimiento de los riesgos residuales y revisar la ejecución de las respuestas a los riesgos mientras se evalúa su efectividad. El proceso supervisión de riesgo, así como los demás procesos de gestión de riesgos, es un proceso continuo que se realiza durante la vida del proyecto.
Otras finalidades del proceso Supervisión de Riesgos son determinar si:

- Las asunciones del proyecto aún son válidas
- El riesgo, según fue evaluado, ha cambiado de su estado anterior, a través del análisis de tendencias
- Se están siguiendo políticas y procedimientos de gestión de riesgos correctos
- Las reservas para contingencias de coste o cronograma deben modificarse para alinearlas con los riesgos del proyecto.

4.4.5. FACTORES CRÍTICOS PARA LA APLICACIÓN DE LA GESTIÓN DE RIESGOS.

- La utilidad de la Gestión de Riesgo es más alta si se la emplea desde las fases iníciales del proyecto.
- La Gestión de Riesgo no tendría ningún efecto si no se hace seguimiento del plan de acción acordado.
- La relevancia de los resultados de la Gestión de riesgo depende de la selección adecuada que se hagan de los ámbitos de indagación importantes para el proyecto en cuestión.
- La Gestión de Riesgo resulta flexible y es posible combinarlo con otras herramientas de análisis.
- Los resultados de la Gestión de Riesgo, se enriquecen en la medida que se entrevista a un número relevante de Stakeholders.

- La Gestión de Riesgo ayuda a formalizar problemas/riesgos y acciones que están solamente a nivel de percepciones.
- La gestión de Riesgo es también una herramienta que vincula las causas con efectos permite idear acciones adecuadas que atacan las raíces de los problemas más que sus síntomas.

4.5.GESTIÓN DE PERSONAL

¿Qué hace el gerente del proyecto con la gestión del personal?
En primer lugar, la creación de un ambiente de trabajo positivo, agradable y productivo, esta es una de las habilidades más importantes de un buen gerente. La buena gestión de empleados se traduce en un personal que trabaja duro, disfruta de sus puestos de trabajo y se mantiene motivado. Un buen gerente puede hacer un trabajo con un alto nivel estrés, en una tarea agradable para sus empleados, y mantener buenos funcionarios a largo plazo. El gerente del proyecto tiene un papel importante no sólo en la gestión de personas, sino también en la creación de un buen ambiente de trabajo para su personal

4.5.1. SELECCIÓN DEL PERSONAL

Características que deben considerarse al momento de seleccionar personal para que cumpla el rol de tester.

Capacidad para "imaginar" soluciones a problemas existentes. Un tester debe ser capaz de poder realizar hipótesis con un grado de fiabilidad aceptable acerca de la causa de un error cuando éste se produce. En esta fiabilidad influye el hecho de que el software debe, por un lado, conocer los principios de arquitectura y diseño que se han seguido en la creación del software, así como tener un dominio bastante amplio de la tecnología con la que ha sido elaborado, del lenguaje, para poder entender el código que compone el sistema y también de los escenarios de uso posibles para dicho software, de modo que sepamos si el error observado corresponde a un caso de uso más o menos frecuente y, a su vez, crítico del sistema.

- Un tester es un **explorador nato**: No tienen ningún miedo por adentrarse en terrenos y situaciones desconocidas. Un buen tester está deseando obtener la última versión beta de un cierto producto para explorar y, a ser posible, ser el primero en encontrarle fallos.
- Un tester es **incansable**: Un tester SIGUE BUSCANDO. Estas palabras resumen la filosofía de explorador de un tester.

Perfeccionistas con criterio: La perfección de un producto radica en el equilibrio perfecto entre fiabilidad y coste. De nada nos sirve un software desarrollado en tiempo récord si tiene muchos fallos; a su vez, tampoco nos resulta útil un software fiable al 99'9999% si el coste de su desarrollo es estratosférico y por sí solo supone un balance negativo superior a las pérdidas que un producto "parcialmente" imperfecto puede conllevar.
Creatividad: Probar lo extraño no es suficiente para un tester. Probar únicamente lo obvio es pecado capital.
- Buen criterio **economista**: Un tester debe ser capaz de decidir qué tipo de bugs buscar, estimar cuánto tiempo debe dedicar a seguir el rastro de un determinado bug o saber

distinguir un bug real de algo que tan sólo es un "comportamiento disperso" de la aplicación.
Hasta aquí, definiríamos las características necesarias para la identificación de bugs.
Diplomacia y capacidad de expresarse con tacto, aún cuando lo que se está haciendo es criticar el trabajo de otros... A pesar de tratarse de una crítica constructiva y enfocada a mejorar el producto final, no es algo sencillo.
Persuasividad: Hay que ser capaz de acotar bien el bug expuesto y, realmente, convencer al resto de miembros del equipo de que dicho fallo DEBE ser resuelto a la mayor brevedad posible.

4.5.2. MOTIVACIÓN

Un ambiente de trabajo positivo es un lugar donde sus empleados desean estar. Muchas empresas están aprendiendo esto y están integrando mejoras para la calidad de vida dentro de los entornos laborales. Como un profesional en gestión de proyectos este es uno de los signos más claros de su éxito. Aprovechar las oportunidades de fomentar el trabajo en equipo entre sus empleados. Si bien llevarlos a todos el próximo fin de semana de aventura al aire libre no será necesario, es importante que se trabaje para animar a la gente a no solo llevarse bien, sino a ayudarse unos a otros.

4.6.GESTIÓN DE CALIDAD

Es el conjunto de actividades de la función empresaria que determina la política de la calidad, los objetivos y las responsabilidades y las implementa por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento de la calidad y el mejoramiento de la calidad, en el marco del sistema de la calidad.

4.6.1. GENERALIDADES DE LA GESTIÓN DE LA CALIDAD

4.6.1.1. DEFINICIÓN DE ISO

Es la International Organization for Standardization (Organización Internacional para la estandarización). La ISO es una red de los institutos de normas nacionales de 157 países que produce las normas para certificar y mejorar los estándares de calidad. La certificación de un Sistema de Gestión de la Calidad es reconocida en el mismo número de países. ISO no es un acrónimo; proviene del griego ISO, que significa igual

4.6.1.2. CERTIFICACIÓN SGC

La certificación es el reconocimiento internacional, al Sistema de Gestión de la Calidad (SGC) con el cual se satisfacen los requisitos cumpliendo las normas de calidad establecidas
El organismo certificador utiliza como marco de referencia los requisitos establecidos en la norma ISO 9001:2000, para determinar la capacidad o efectividad del Sistema de Gestión de la Calidad de la empresa interesada en obtener la certificación.
En consecuencia se debe establecer, documentar, implementar y mantener un SGC y mejorar continuamente su eficacia de acuerdo con la Norma ISO 9001:2000.

4.6.1.2.1. Qué acciones deben llevarse a cabo para lograr la certificación?

1. Elaborar diagnóstico general de la organización
2. Elaborar diagnóstico de la Calidad actual
3. Definir la política de calidad
4. Definir los objetivos de mejoramiento de la Calidad
5. Elaborar Mapa de procesos
6. Documentar procesos
7. Elaborar Manual de Calidad
8. Realizar Auditorias
9. Ajustar el sistema de Gestión de la Calidad mediante correcciones, acciones correctivas, preventivas y de mejoramiento.
10. Adelantar revisiones por parte de la dirección y
11. Obtener y mantener la certificación.

4.6.1.3. QUÉ ES UN SISTEMA DE GESTIÓN DE CALIDAD?

Es el conjunto de procesos interrelacionados que transforman las entradas o insumos en salidas o resultados que satisfacen los requisitos o expectativas de los clientes. Las entradas principales del sistema son los requisitos o expectativas de quienes pagan o son beneficiarios de productos y servicios, y las salidas son los mismos productos o servicios que satisfacen a los clientes.

4.6.1.4. QUÉ ES GESTIÓN?

Es el conjunto de “Actividades coordinadas para dirigir y controlar una organización” Numeral 3.2.6 Norma ISO 9000-2000.

Las organizaciones se dirigen y controlan en relación con sus planes, procesos, metas y propósitos (MISIÓN, VISIÓN, OBJETIVOS).

Según la Norma ISO 9001:2000 se debe “Implementar las acciones necesarias para alcanzar los resultados planificados y la mejora continua de los procesos”.

4.6.1.5. QUÉ ES UN PROCESO?

La norma ISO 9000 define un proceso como un "conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados"

Una organización es un conjunto de procesos. La calidad y eficacia de los procesos determina la calidad de los productos y servicios.

CAPITULO 5.- Estándares internacionales de Calidad

5.1. ¿QUÉ ES EL CMM - CMMI?

El Capability Maturity Model Integration (CMMI) es un marco de referencia que las organizaciones emplean para mejorar sus procesos de desarrollo. El CMMI nació en el Software Engineering Institute perteneciente a la Carnegie Mellon University,

El CMM - CMMI es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software.

Basados en los principios de la calidad total (TQM) popularizados por autores como Crosby, Deming y Juran, estos modelos proponen un conjunto de prácticas que las organizaciones pueden adoptar para implantar procesos productivos más efectivos

5.1.1. El nacimiento de CMM - CMMI

El departamento de defensa de los Estados Unidos tenía muchos problemas con el software que encargaba desarrollar a otras empresas, los presupuestos se disparaban, las fechas alargaban más y más.

Como esta situación parecía intolerable se convocó a un comité de expertos para que solucionaran estos problemas, en el año 1983 dicho comité concluyó "Tienen que crear un instituto de la ingeniería del software, dedicado exclusivamente a los problemas del software, y a ayudar al Departamento de Defensa".

El SEI (Software Engineering Institute) es el instituto que creó para que solucionaran los problemas antes mencionados

5.1.2. Niveles CMM – CMMI

Los niveles CMM - CMMI son 5:

Inicial o Nivel 1 CMM - CMMI. Este es el nivel en donde están todas las empresas que no tienen procesos. Los presupuestos se disparan, no es posible entregar el proyecto en fechas las fechas establecidas, los empleados prolongan sus jornadas de trabajo, para terminar un proyecto. No hay control sobre el estado del proyecto, el desarrollo del proyecto es completamente opaco, los ejecutivos no conocen absolutamente nada de lo que ocurre en el proyecto.

Repetible o Nivel 2 CMM - CMMI. Quiere decir que el éxito de los resultados obtenidos se puede repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión de requisitos
- Planificación de proyectos
- Seguimiento y control de proyectos
- Gestión de proveedores

- Aseguramiento de la calidad
- Gestión de la configuración

Definido o Nivel 3 CMM - CMMI. Alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) está totalmente definida, es decir que está establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.

Los procesos que hay que implantar para alcanzar este nivel son:

- Desarrollo de requisitos
- Solución Técnica
- Integración del producto
- Verificación
- Validación
- Desarrollo y mejora de los procesos de la organización
- Definición de los procesos de la organización
- Planificación de la formación
- Gestión de riesgos
- Análisis y resolución de toma de decisiones

Gracias a los beneficios proporcionados por el nivel tres, algunas empresas no ven la necesidad de evolucionar hacia un nivel cuatro o cinco, dado que el nivel tres se adapta perfectamente a sus requerimientos y necesidades del negocio.

Cuantitativamente Gestionado o Nivel 4 CMM - CMMI. Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión cuantitativa de proyectos
- Mejora de los procesos de la organización

Optimizado o Nivel 5 CMM - CMMI. Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.

Los procesos que hay que implantar para alcanzar este nivel son:

- Innovación organizacional
- Análisis y resolución de las causas

Normalmente las empresas que intentan alcanzar los niveles 4 y 5 lo realizan simultáneamente ya que están muy relacionados.

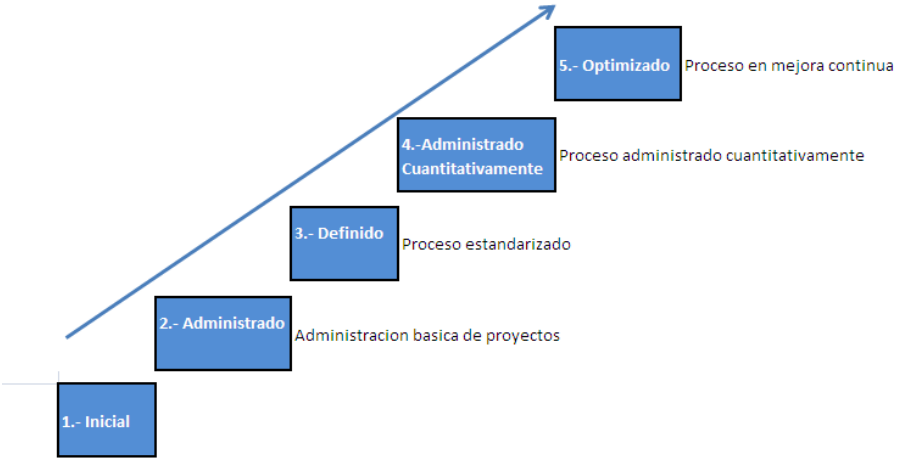


Figura 006.- Niveles de Madurez CMMI

5.1.3. Beneficios de CMMI

La correcta aplicación del modelo trae beneficios como:

5.1.3.1. Beneficios puramente "ingenieriles":[22]

1. Los procesos maduros permiten:
   - Entender lo que está pasando.
   - Que el personal desarrolle todo su potencial más completamente y más efectivamente dentro de la organización.
2. La mejora de los procesos tiene más posibilidades de resultar con éxito y ser más beneficiosa para la organización ya que se basa en la definición, medición y control de los procesos.
3. Se incrementa notablemente la probabilidad de éxito en la introducción de nuevas y apropiadas tecnologías, técnicas y herramientas en la organización.

5.1.3.2. Beneficios "económicos" u "organizativos"[22]

1. Enfatiza el desarrollo de procesos en las organizaciones que permiten mejorar el desarrollo de los productos y los servicios ofertados a los clientes.
2. Proporciona un marco de trabajo que permite organizar y priorizar las actividades de mejora de procesos, involucrando al propio producto, al negocio, al personal y a la tecnología.
3. Da soporte a la coordinación de actividades multidisciplinarias que pueden ser necesarias para construir con éxito un determinado producto.
4. Enfatiza el alineamiento de los objetivos de la mejora de procesos con los objetivos de negocio de las organizaciones

[22] [VILLA. M]

5.2. TMMI (Test Maturity Model Integrated)

El TMMI proporciona un enfoque más amplio y estructurado al proceso de prueba, existe un gran número de experiencias positivas con el modelo TMMI y las organizaciones que la utilizan van en aumento, las experiencias muestran que aplicar el modelo de madurez TMMI tiene un impacto positivo sobre la calidad del producto.

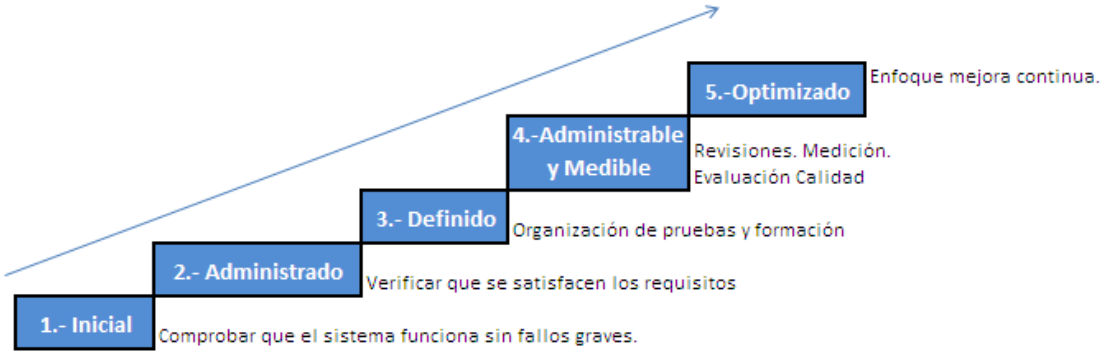


Figura 007.- Los 5 niveles de Tmmi.

El TMMi no es sólo un modelo teórico, sino un conjunto bien definido de áreas proceso y objetivos basados sobre casos de estudio prácticos. Aunque el modelo es bastante fácil de entender, su aplicación en una organización no siempre es una tarea simple. En promedio se puede tardar 2 años para alcanzar el TMMI nivel 2.

Nivel 1

En TMMI nivel 1, las pruebas son un caos, y el proceso definido a menudo es considerado una parte de la depuración.

La organización no suele proporcionar un entorno estable para apoyar los procesos. El éxito en estas organizaciones depende de la competencia y el compromiso de la gente en la organización y no del uso de procesos.

Las pruebas se desarrollan en una manera ad hoc después de que la programación a finalizado. Pruebas y la depuración son intercalados para obtener los errores del sistema. El objetivo de las pruebas de este nivel es mostrar que el software se ejecuta sin mayores fracasos.

Los productos son liberados sin la visibilidad adecuada con respecto a la calidad y los riesgos. Ya en la realidad, el producto no suele cumplir con sus necesidades, no es estable, o es demasiado lento para trabajar. TMMi en el nivel 1 no hay ningún proceso definido. Las organizaciones de nivel 1 se caracterizan por una tendencia a cometer más, el abandono de los procesos en un momento de crisis, y la incapacidad para repetir sus éxitos. También tienden a no entregar el producto a tiempo, sobrepasar los presupuestos y la calidad no está de acuerdo con las expectativas del cliente.

El plan de ensayo se define lo que se requiere la prueba, cuándo, cómo y por quién. Se establecen compromisos con las partes interesadas y revisado según sea necesario. Las pruebas son vigiladas y controladas para asegurar que va de acuerdo con el plan y las acciones se pueden tomar si se producen desviaciones. El estado de la labor de productos y la prestación de servicios de pruebas son visibles a la gestión. Y para obtener la selección

de casos de prueba de las especificaciones técnicas de diseño del ensayo se aplican. Sin embargo, todavía puede iniciar la prueba relativamente tardía en el desarrollo del ciclo de vida, por ejemplo, durante el diseño, o incluso durante la fase de codificación. Las pruebas son multi-nivel: hay unidad, la integración, la aceptación del sistema y niveles de prueba. Para cada prueba de nivel hay pruebas de los objetivos específicos definidos en el nivel de toda la organización o programa de pruebas en toda la estrategia.

Nivel 2

En TMMi nivel 2, las pruebas se convierte en un proceso gestionado y está claramente separada de la depuración. El proceso de la disciplina se refleja en el nivel 2 de madurez y contribuye a garantizar que las prácticas actuales se mantienen durante momentos de estrés. Sin embargo, las partes interesadas todavía perciben a las pruebas como una fase del proyecto que sigue la codificación. En el contexto de la mejora del proceso de prueba, la estrategia de pruebas se encuentra establecida y definida a lo largo de la empresa y del programa de pruebas. El Plan de prueba también se está instituyendo. Dentro del plan de prueba el enfoque se basa en el resultado de un producto de evaluación de riesgos. Técnicas de gestión de riesgos se utilizan para identificar los riesgos de los productos basados en las necesidades documentadas.

En el plan de pruebas, se define lo que se requiere de la prueba, cuándo, cómo y por quién se va a realizar.
Se establecen compromisos con las partes interesadas y se revisan según sea necesario. Las pruebas son vigiladas y controladas para asegurar que va de acuerdo con el plan y las acciones que se pueden tomar si se producen desviaciones.

El principal objetivo de las pruebas en un TMMi nivel 2, es verificar que el producto cumple los requisitos especificados. El propósito también es distinguir claramente los procesos de prueba y depuración. Muchos problemas de calidad en este nivel se produce porque las pruebas generan defectos que se han propagado de los requisitos y el diseño en código hasta las fases finales en el desarrollo del ciclo de vida sin embargo en este nivel todavía no hay programas formales de revisión para hacer frente a esta importante cuestión.

El proceso en las zonas TMMi nivel 2 son:
2.1Política y Estrategia de pruebas
2.2 Planificación de las pruebas
2.3 Seguimiento y Control de las pruebas
2.4 Diseño y Ejecución de pruebas
2.5 Pruebas de Ambiente

Nivel 3

En TMMi nivel 3, La fase de pruebas no se la ejecuta luego de la codificación. Las pruebas están plenamente integradas en la fase de desarrollo. Prueba de la planificación se realiza en una primera etapa del proyecto. El desarrollo de un plan maestro de prueba se basa en la prueba de la planificación de las competencias adquiridas en el nivel 2. La organización del

conjunto de procesos, que es la base para el nivel 3 de madurez, se ha establecido y mejorado con el tiempo.se implementan herramientas básicas de apoyo principales a las actividades de pruebas se recopilan casos de prueba, almacenan y gestionan en una base de datos central para la reutilización y pruebas de regresión. Organizaciones de este nivel comienzan a darse cuenta de la importancia de las revisiones en el control de calidad; una revisión formal del programa se aplica, aunque aún no vinculados a la dinámica del proceso de pruebas

El proceso en las zonas TMMi nivel 3 son las siguientes:
3.1 Prueba de Organización
3.2 Programa de Formación de prueba
3.3 Prueba de integración al ciclo de vida
3.4 No pruebas Funcionales
3.5 revisiones inter pares

Nivel 4
En TMMi 4 la organización de pruebas es un proceso bien definido, bien fundado y mensurable. En el nivel 4 de madurez, la organización y los proyectos para establecer objetivos cuantitativos de calidad de los productos y el rendimiento de los procesos y los utilizan como criterios en el manejo de ellos. La calidad del producto y el rendimiento del proceso se entienden en términos estadísticos y se gestiona todo el ciclo de vida. Medidas se incorporen en la organización del repositorio de apoyo a la medición basada en hechos la toma de decisiones.
Los comentarios y las inspecciones se consideran parte de las pruebas y de documento utilizado la medida de calidad.
Las pruebas dinámicas y estáticas se integran en un solo enfoque. Los comentarios están formalmente utilizados como medio de control de calidad. Los productos se evaluarán por medio de criterios cuantitativos de los atributos de calidad tales como fiabilidad, facilidad de uso y facilidad de mantenimiento. Una organización con un programa amplio de pruebas de medición proporciona información y la visibilidad con respecto al proceso de prueba.
Se percibe como las pruebas de evaluación, que consiste en el ciclo de vida de todas las actividades relacionadas con la comprobación de los productos y los productos relacionados con el trabajo.

El proceso en las zonas TMMi nivel 4 son los siguientes:
4.1 Test de medición
4.2 La evaluación de calidad de los productos
4.3 Revisiones Avanzadas de pares

Nivel 5
Sobre la base de todos los resultados que se han logrado mediante el cumplimiento de todos los objetivos de mejora de los anteriores niveles de madurez, la prueba es ahora un proceso completamente definido y es capaz de controlar los costes y la eficacia de las pruebas. En TMMi nivel de madurez 5, una organización que mejora continuamente los procesos sobre la base de una comprensión cuantitativa de la causa común de variación inherente en los procesos. Mejorar el rendimiento del proceso de prueba se lleva a cabo a través de proceso

gradual e innovadoras y mejoras tecnológicas. Los métodos y técnicas están optimizados y se centrará en un continuo de ajuste y prueba de la mejora de procesos. Defecto de prevención y control de calidad se practican. Estadísticos de muestreo, las mediciones de los niveles de confianza, confianza, fiabilidad y conducir el proceso de prueba. Entre otros "Prevención de defectos" y "Control de Calidad" se introdujo como proceso zonas.
El proceso de prueba se caracteriza por la toma de muestras de calidad basados en las mediciones. Un detallado procedimiento de selección y evaluación de instrumentos de prueba. Instrumentos de apoyo para el proceso de prueba lo más posible durante el ensayo de diseño, ejecución de pruebas, pruebas de regresión, la prueba de manejo de casos, etc.
Proceso de reutilización también se practica en el nivel 5 el apoyo de un proceso activo biblioteca. Pruebas es un proceso con el objetivo de prevenir defectos.

Áreas en proceso de nivel 5 son:
5.1 Prevención de defectos
5.2 Optimización de procesos de prueba
5.3 Control de Calidad

5.3. ¿QUÉ SON LAS NORMAS ISO?

Un conjunto de normas internacionales genéricas que establecen sistemas de gestión de la calidad aplicados por organizaciones de cualquier tipo o tamaño que fabrican productos o componentes (hardware), fabrican software, fabrican materiales procesados, ofrecen servicios, desempeñan funciones de administración pública.

5.3.1. ¿Qué no son las normas ISO?

- No son Especificaciones de Calidad de Productos.
- No son obligatorias.
- No es un programa de corta duración.
- No es el punto final de la mejora continua
- El modelo de un sistema de gestión de la calidad basado en procesos que se muestra a continuación ha servido a las organizaciones para enfocar sus esfuerzos en aquellos procesos que aportan valor al cliente y garantizan la satisfacción de sus necesidades declaradas e implícitas:

El proceso de Responsabilidad de la Dirección se identifica con los procesos estratégicos que aportan directivas al resto de los procesos, tales como la definición de políticas, objetivos, responsabilidad y autoridad, comunicación, así como el compromiso de la dirección y la revisión del sistema por parte de ésta.
La gestión de recursos se refiere a la infraestructura y los recursos humanos, materiales y financieros que se identifica con los procesos de apoyo que soportan los procesos de realización, donde se manifiesta la cadena de valor que transforma las necesidades y expectativas de los clientes en productos que cumplan con sus requerimientos.
Luego la medición, análisis y mejora permitirá determinar la eficacia, eficiencia y efectividad del resto de los proceso, y aportará la información necesaria para la toma de decisiones y la mejora continua del producto, los procesos y el sistema.

Es importante no ver estos procesos en orden cronológico ni como correspondientes a diferentes partes de la estructura de la organización, sino que son procesos simultáneos, íntimamente relacionados y extensivos a toda la organización.

Las normas ISO 9000:2000 fueron aprobadas en diciembre del año 2000 y constan de un grupo de 3 normas individuales, pero relacionadas entre sí, enfocadas a la administración de la calidad y el aseguramiento de la calidad

Familia	Definición
ISO 9000:2000	Sistemas de Gestión de la calidad fundamentos y vocabulario
ISO 9001:2000	Sistemas de Gestión de calidad, requisitos(única norma certificable)
ISO 9004:2000	Sistemas de Gestión de Calidad, directrices para la mejora del desempeño

Figura 008.- Cuadro con definiciones de las familias ISO900:

La norma ISO 9001:2000 e ISO 9004:2000 tiene una estructura común basada en cuatro pilares fundamentales:

- Responsabilidad de la Dirección
- La gestión de recursos
- Realización del producto
- Medición, análisis y mejora

5.3.2. ISO 9001:2000

"El ISO 9001:2000, es un estándar internacional que puede ser aplicado a cualquier tipo de organización. Este estándar, que establece los requerimientos que debe cumplir un sistema genérico de gestión de la calidad (QMS, por sus siglas en inglés), puede ser usado para mejorar procesos internos, para obtener una certificación o para establecer relaciones contractuales. Si bien su ámbito de aplicación es más amplio, existen lineamientos para su uso en organizaciones de sistemas."[23]

Consta de 20 "políticas", que son los parámetros sobre los cuales se basan las diferentes áreas de su sistema de calidad.
Es la Norma más completa y fue diseñada para Empresas que diseñan, producen y venden productos o servicios.

"Las cinco secciones en que se divide ISO 9001:2000 son:

1. QMS Sistema de Gestión de la Calidad (Requisitos generales y Requisitos de la documentación).
2. Responsabilidad de la Gestión (Compromiso de la dirección, Enfoque al cliente, Política de la calidad, Planificación).
3. Gestión de los Recursos (Provisión de recursos, Recursos humanos, Infraestructura, Ambiente de trabajo).

[23] [ISO 9001:2000]

4. Realización del Producto (Planificación de la realización del producto, Procesos relacionados con los clientes, Diseño y desarrollo, Compras, Prestación del servicio).
5. Medición, Análisis y Mejora (Generalidades, Supervisión y Medición, Control de servicio no-conforme, Análisis de datos, Mejora)." [24]

5.3.2.1. La Norma ISO 9001.

ISO 9001:1994 tenía requisitos:
Responsabilidad de la Dirección.

- Sistema de la Calidad.
- Revisión de Contratos.
- Control del Diseño.
- Control de los Documentos y los Datos.
- Compras.
- Control de Productos Suministrados por el Cliente.
- Identificación y Trazabilidad del Producto.
- Control de los Procesos.
- Inspección y Ensayos.
- Control de los Equipos de Inspección, Medición y Ensayo.
- Estado de Inspección y Ensayo.
- Control de Productos no Conformes.
- Acciones Correctivas y Preventivas.
- Manipuleo, Almacenamiento, Embalaje, Conservación y Entrega.
- Control de los Registros de la Calidad.
- Auditorias Internas de la Calidad.
- Capacitación.
- Servicios Postventa.
- Técnicas Estadísticas.

Esta norma estuvo vigente hasta diciembre del 2003.
La organización ISO exige una revisión de sus normas de forma periódica, para asegurar que son actuales y satisfacen las necesidades de los usuarios, así entonces y luego de la revisión se tiene una nueva versión de la norma, destacando los siguientes puntos.

1. Medir la Satisfacción del Cliente
2. Mejora Continua
3. Gestión de los Recursos
4. Gestión del Proceso

Los principales cambios que presenta la norma luego de la revisión, son los siguientes:

- Estructura orientada a proceso y una secuencia más lógica de los contenidos
- Mejora Continua
- Medida de la Satisfacción del Cliente como indicador de Mejora
- Atención a los recursos tales como comunicación y entorno de trabajo
- Mejoras en la terminología para una interpretación más fácil
- Compatibilidad con el Sistema de Gestión Ambiental

[24] [VILLA. M]

- Referencia específica a principios de gestión de calidad
- Autoevaluación como medio de mejora (ISO 9004).

Además se incluyen los siguientes requisitos:

- Medida Satisfacción del Cliente
- Mejora Continua
- Disponibilidad de Recursos
- Mediciones extensivas a sistema, procesos y producto (incluido servicio)
- Análisis del cumplimiento y eficacia del Sistema de Calidad

5.3.2.2. Estrategia para la aplicación de la ISO 9000:2000

Es común que las empresas que acometen la implantación de sistemas de calidad ISO 9000 se enfrenten a las siguientes situaciones:

Se contratan consultores externos que aunque conozcan la calidad a nivel gerencial desconocen las especificidades de los procesos de software. Esto dificulta la adaptación eficaz de la norma a la organización y afecta la eficiencia del proceso de implantación.

El exceso de documentación provoca un rechazo general a la aplicación de la norma.

Se pretende comenzar la implantación de arriba hacia abajo, de modo que transcurre mucho tiempo en llegar al proceso concreto de realización del producto. Esto hace que no se vean resultados a corto plazo.

Se da más importancia a las cuestiones organizativas de la administración que a las del propio proyecto.

No se consideran las especificidades del proceso productivo como proceso de conocimiento: difícil estandarización, intervención del cliente prácticamente durante todo el proceso, mayor importancia del servicio posventa.

5.4. SPICE (SOFTWARE PROCESS IMPROVEMENT AND CAPABILITY DETERMINATION)

SPICE es una norma que trata los procesos de ingeniería, gestión, relación cliente-proveedor, de la organización y del soporte.

Fue creada por la alta competencia del mercado de desarrollo de software, a la difícil tarea de identificar los riesgos, cumplir con el calendario, controlar los costos y mejorar la eficiencia y calidad. Este engloba un modelo de referencia para los procesos y sus potencialidades sobre la base de la experiencia de compañías grandes, medianas y pequeñas.
ISO/IEC 15504 es un emergente estándar internacional de evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería del software, con la filosofía de desarrollar un conjunto de medidas de capacidad estructuradas para todos los procesos del ciclo de vida y para todos los participantes. Es el resultado de un esfuerzo internacional de trabajo y colaboración y tiene la innovación, en comparación con otros modelos, del proceso paralelo de evaluación empírica del resultado.

Descripción del modelo:
El modelo describe los procesos que una organización puede ejecutar, adquirir, suplir, desarrollar, operar, evolucionar, brindar soporte de software y todas las prácticas genéricas que caracterizan las potencialidades de estos procesos.

La arquitectura se basa en:
Prácticas base: Son las actividades esenciales de un proceso específico, agrupado por categorías de procesos de acuerdo al tipo de actividad que direccionan.
Prácticas genéricas: Aplicables a cualquier proceso, que representa las actividades necesarias para administrar el "proceso" y mejorar su potencialidad.

- **Cliente Suministrador**.- consta del proceso que directamente impacta en el cliente, soportan el desarrollo, la entrega, al cliente del Software proporcionándole su correcta operación y utilización
- **Ingeniería.**- consta de procesos que especifican implementan o mantienen un sistema, el producto de SW y su documentación de usuario
- **Soporte.**- proporcionan un proceso de soporte que puede ser empleado en cualquiera de las categorías anteriores su objetivo brindar soporte en las actividades que realiza
- **Gestión.**- son procesos que constan de prácticas que tienen una naturaleza genérica para gestionar cualquier clase de proyecto de SW
- **Organización.**- procesos que establecen los objetivos del negocio de la organización y del producto del proceso de desarrollo ayuda a cumplir de buena forma con los objetivos del negocio

	ISO 9001:2000	CMMI	ISO 15504-SPICE
Ámbito de aplicación	Genérico	Software y Sistemas	Software y Sistemas
En su favor	El más extendido y sencillo	El de mayor prestigio	Más consensuado y probado
En su contra	Simple, general, no guía paso a paso	Difícil de entender, mayor inversión, prescriptivo	Difícil en capacidad, complejo para evaluar
Procesos	Estructura propia	Estructura propia	Delega en ISO 12207, por mayor aplicabilidad
Validación	Encuestas satisfacción	Encuestas satisfacción y casos de estudio	'Trials' y esfuerzo empírico
Objetivo	Cumplimiento de requisitos de calidad por procesos	Mejora del proceso, determinación capacidad contratista	Valoración del proceso y guía para la mejora.
Representación	Plana	Continua y por etapas	Continua (por etapas a nivel de proceso)
Técnicas análisis	Guías y listas de comprobación	Cuestionarios de evaluación	Varios

Figura 009.- Tabla Comparativa entre modelos analizados [25]

[25] [VILLA. M]

CAPITULO 6.- EL COSTE DE LA CALIDAD VS EL COSTO DE LA NO CALIDAD

6.1. DEFINICIÓN DEL COSTO DE LA [MALA] CALIDAD DEL SOFTWARE

Si la calidad no tiene costos asociados, cuál es la razón de tener un programa de aseguramiento de calidad. Ya se conoce que la mala calidad sí tiene costos, tanto por el costo de corrección como por los problemas que acarrean nuestros clientes, finalmente un cliente descontento por la falta de calidad en nuestros productos, fácilmente puede dejar de ser nuestro cliente.

No hay visión uniforme de lo que es costo de calidad y lo que debe ser incluido bajo este término. Las ideas acerca del costo de calidad han venido evolucionando rápidamente en los últimos años. Antes se tomaba en cuenta costos en calidad al costo de poner en marcha el departamento de aseguramiento de la calidad, la detección de costos de desecho y costos justificables.
Actualmente, se entienden como costos de calidad aquéllos incurridos en el diseño, implementación, operación y mantenimiento de los sistemas de calidad de una organización, aquéllos costos de la organización comprometidos en los procesos de mejoramiento continuo de la calidad, y los costos de sistemas, productos y servicios frustrados o que han fracasado al no tener en el mercado el éxito que se esperaba.

"El costo de la calidad tiene dos componentes: lo que invertimos en obtener buena calidad y lo que pagamos por no lograrla. La primera componente es decidida por nosotros y controlada; la segunda no la decidimos sino que se manifiesta en las fallas de nuestro producto.
Invertimos en tener buena calidad mediante *prevención* (evitar errores) y *evaluación* (verificar que no tenemos errores). Por otro lado, las fallas pueden ser de dos tipos: internas (las que encuentran los desarrolladores) y externas (las que encuentran los clientes)." [26]

6.2. CÓMO MEDIR EL COSTE DE LA CALIDAD EN FORMA EFICIENTE

La forma estándar de medir el costo de la calidad en el desarrollo de software "Es registrar el esfuerzo en horas dedicadas a actividades asociadas a la calidad. Esto es muy barato si ya se cuenta con un sistema de reporte de horas de proyectos."[28]
Existe una relación entre costos, calidad, inversiones y mejoramiento de la calidad. De ahí que la clasificación de costos más utilizada esté referida fundamentalmente a tres categorías: prevención, valoración o cuantificación y fallas/fracasos.
Las ventajas de esta particular categorización son, primeramente que están universalmente aceptadas; segundo, cubre la mayoría de las clases de costos, y tercero, la más importante,

[26] [STRAUB. P]

suministra un criterio generalizado que ayuda a precisar de qué costo se trata, en donde se ubica y si es relacionado con la calidad.
Con el propósito de favorecer un acercamiento mayor a las decisiones de negocios, a esta clasificación, se han sumado otros elementos a ponderar, como son: los proveedores, la propia empresa y los consumidores.
Muchos de los costos postventa y postgarantía, pueden ser incluidos bajo estos rubros.
Estas clasificaciones son enunciativas, más no exhaustivas, ya que los costos de calidad siempre estarán en función del propósito al que responden. En este sentido lo recomendable es que los costos que se identifiquen propicien la acción y la toma de decisiones que deriven en el mejoramiento continuo especialmente de los productos, procesos, servicios y proveedores.

6.2.1. Categorías del costo de la calidad

***Prevención*:** Son las actividades para evitar correcciones (ejemplos, entrenamiento asociado al proyecto, mejoras de procesos para el proyecto, recolección
y análisis de métricas).
***Evaluación*:** Son las actividades para determinar la calidad de los entregables (ejemplos, crear casos de prueba, inspecciones de código, auditorías).
***Fallas internas*:** Son las actividades para corregir entregables (ejemplos, modificaciones a un documento después de una inspección, debugging).
***Fallas externas*:** actividades para corregir productos ya entregados (ejemplos, corrección de defectos de productos instalados, correcciones de datos).
***Creación*:** Son las actividades para hacer los entregables del proyecto (ejemplos, levantamiento de requerimientos, codificación).
***Otro*:** actividades no relacionadas con los entregables del proyecto (ejemplos, reporte del estado del proyecto, actividades administrativas).

6.3. COSTOS DE VALORACIÓN O CUANTIFICACIÓN DE LA CALIDAD

Se incurre en estos costos al realizar: inspecciones, pruebas y otras evaluaciones planeadas que se usan para determinar si lo producido, los programas o los servicios cumplen con los requisitos establecidos. Se incluyen especificaciones de mercadotecnia y clientes, así como los documentos de ingeniería e información inherente a procedimientos y procesos. Son elementos específicos los siguientes:

- Inspección y prueba de prototipos.
- Análisis del cumplimiento con las especificaciones.
- vigilancia de proveedores.
- Inspecciones y pruebas de recepción.
- Actividades para la aceptación del producto.
- Aceptación del control del proceso.
- Inspección de embarque.
- Estado de la medición y reportes de progreso.

6.4.COSTOS DE FALLA O FRACASO

Están asociados con cosas que no se ajustan o que no se desempeñan conforme a los requisitos, así como con los relacionados con incumplimientos de ofrecimientos a los consumidores, se incluyen todos los materiales y mano de obra involucrada. Puede llegarse hasta rubros relativos a la pérdida de confianza del cliente. Los rubros específicos son:

- Asuntos con el consumidor (reclamaciones, demandas, atención de quejas, negociaciones, etc.).
- Rediseño.
- Ordenes de cambio para Ingeniería o para Compras.
- Costos de reparaciones.
- Aplicación de garantías.

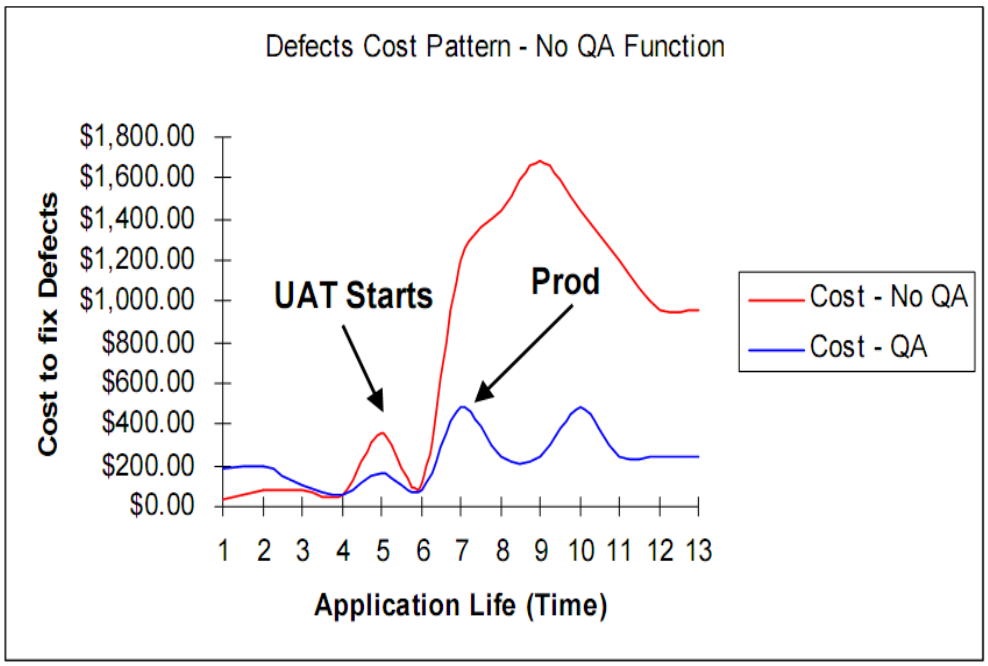


Figura 010.- Costos de las fallas del Software [27]

Este imagen es una comparación entre una aplicación que ha sido desarrollada por un equipo de desarrollo tomando en cuenta la calidad del SW (línea azul) y que ha sido puesta en producción.
Por otro lado la línea roja muestra una aplicación que ha sido desarrollada por un equipo sin tomar en cuenta la calidad del SW y que ha sido puesta en producción.
Los costos que ocasiona corrección de defectos en la etapa de producción ya ocasiona una perdida como tal pero la perdida se hace mucho mas grande cuando el desarrollo ejecutado no tomo en cuenta la calidad del SW, esto nos indica claramente la importancia que se debe dar al desarrollo de software teniendo en cuenta principios de SQA.

[27] [BOEHM. B]

CAPITULO 7.- ANÁLISIS DEL NIVEL DE CALIDAD DE LAS EMPRESAS QUE DESARROLLAN SOFTWARE

7.1.INTRODUCCIÓN

Con el objetivo de conocer el nivel de calidad con el que trabajan las empresas desarrolladoras de software, se administró un cuestionario, básico de 14 preguntas, este cuestionario se ejecuto en 7 empresas ubicadas en la ciudad de Quito.

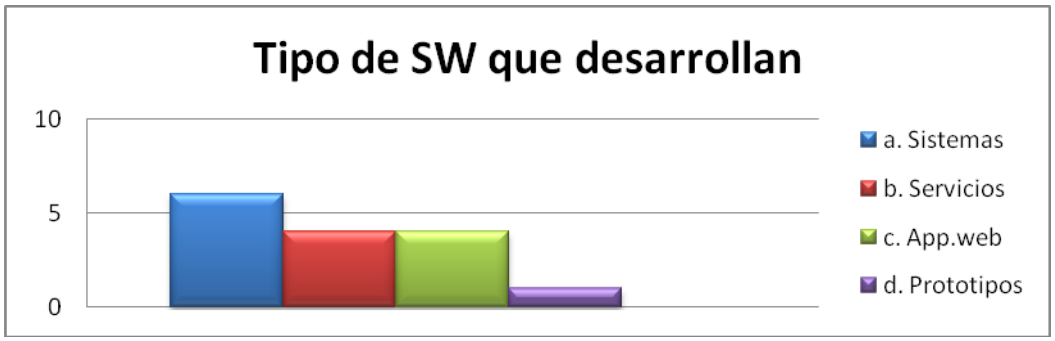
Para probar la validez del contenido, se sugirió que los responsables de contestar este cuestionario tengan perfil de tipo técnico especializado (arquitectos de software), y personal involucrado con la gestión del proceso de desarrollo de software (lideres de proyecto, gerentes de proyecto). En general un grupo de expertos en el área de computación con una experiencia mínima de 3 años, quienes nos proveerán de respuestas adecuadas y realmente confiables, haciendo que el análisis cuente con un nivel de veracidad considerable.

Para ver el contenido de la encuesta. *Ver Anexo 005*

7.2.PRESENTACIÓN DE RESULTADOS

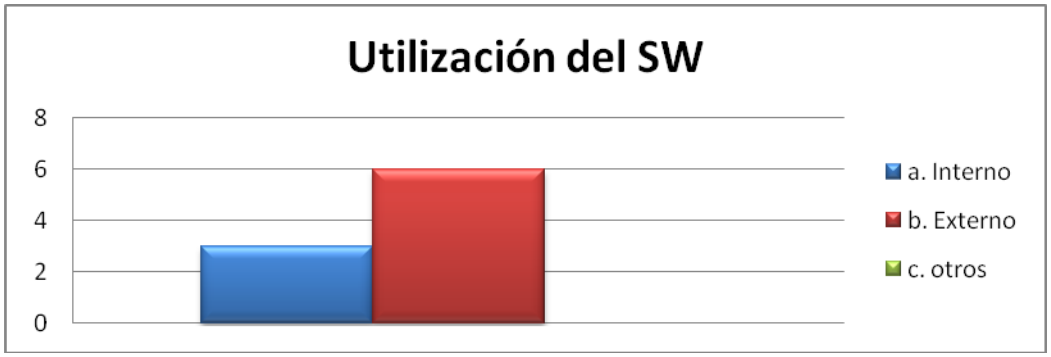
1. El software que desarrollan consiste de:

	Unidades	Porcentaje
a. Sistemas	6	40,00
b. Servicios	4	26,67
c. App.web	4	26,67
d. Prototipos	1	6,67
e. Otros:		



2. El software que desarrollan es para uso:

	Unidades	Porcentaje
a. Interno	3	33,33
b. Externo	6	66,67
c. otros		

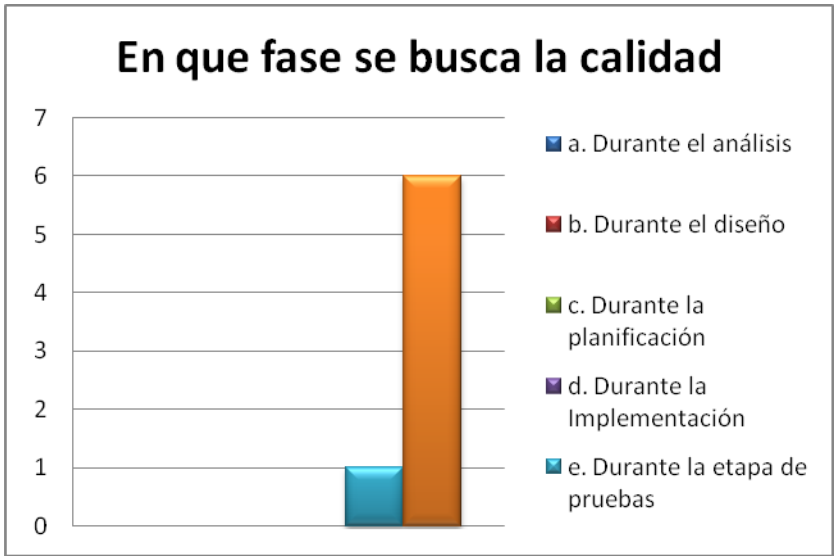


3. Qué papel desempeña la calidad en el software que se desarrolla, categorizando el nivel de calidad que se requiere, se puede decir que la calidad :

	Unidades	Porcentaje
a. si es importante	7	100
b. es algo importante		
c. no es nada importante		

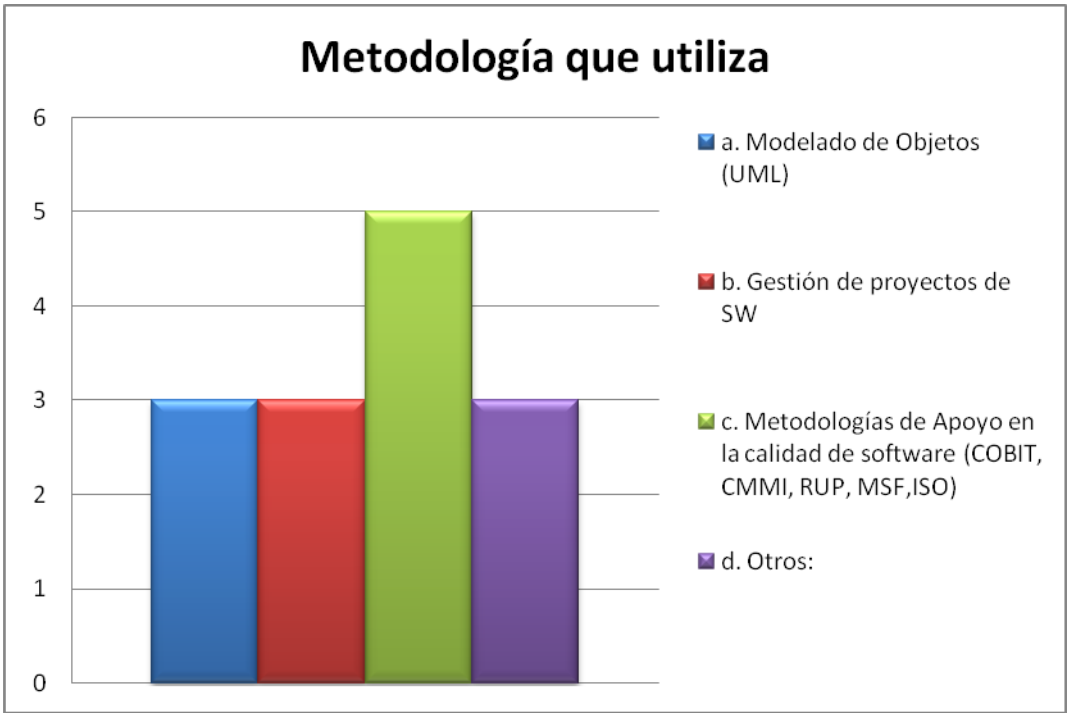
4. En qué fase del ciclo de vida del SW se busca la calidad del software:

	Unidades	Porcentaje
a. Durante el análisis		
b. Durante el diseño		
c. Durante la planificación		
d. Durante la Implementación		
e. Durante la etapa de pruebas	1	14,29
f. Todo el ciclo de vida	6	85,71



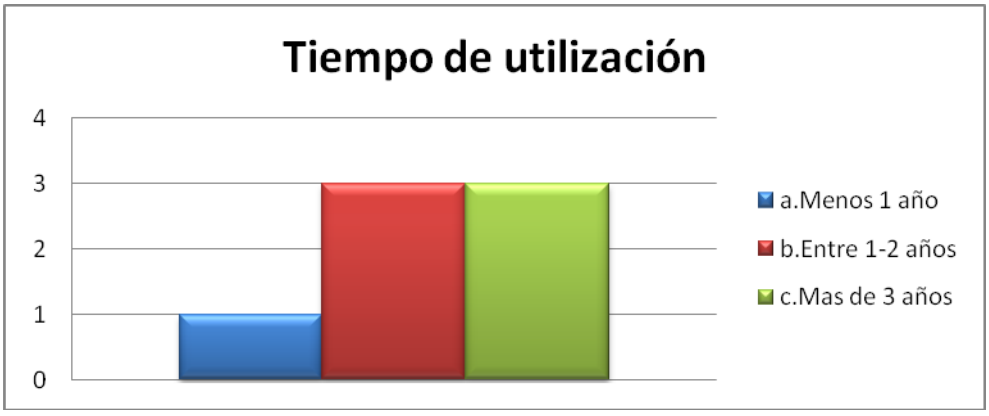
5. Que metodología, procedimientos o lineamientos metodológicos de apoyo en la calidad SW, utiliza su organización para el desarrollo de SW

	Unidades	Porcentaje
a. Modelado de Objetos (UML)	3	21,43
b. Gestión de proyectos de SW	3	21,43
c. Metodologías de Apoyo en la calidad de software (COBIT, CMMI, RUP, MSF,ISO)	5	35,71
d. Otros:	3	21,43



7. Qué tiempo ha utilizado dicho procedimiento, metodología… para el desarrollo de SW en su organización :

	Unidades	Porcentaje
a.Menos 1 año	1	14,29
b.Entre 1-2 años	3	42,86
c.Mas de 3 años	3	42,86

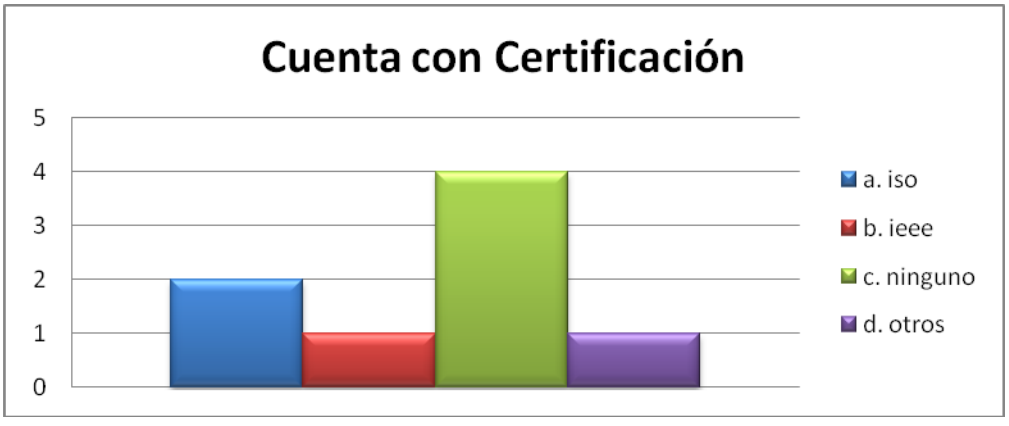


8. Considera importante utilizar una metodología de desarrollo de SW, que garantice, los mínimos estándares de calidad :

	Unidades	Porcentaje
a. Muy importante	7	100
b. Importante		
c. Nada importante		

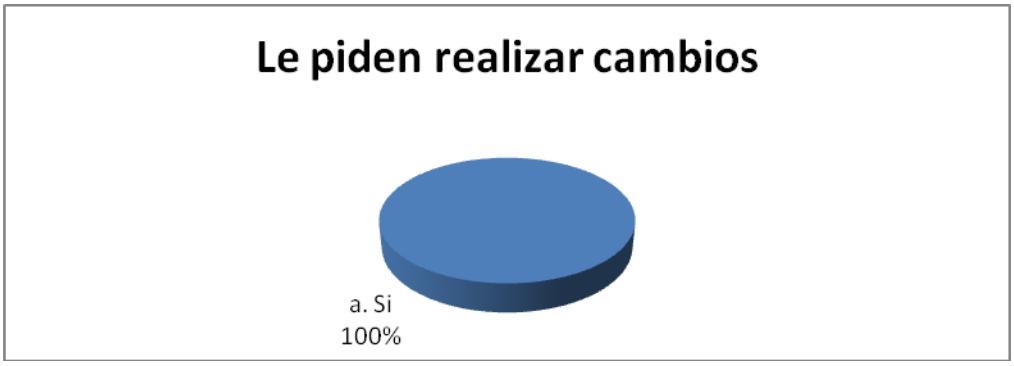
9. El departamento o área de sistemas que desarrolla SW, cuenta con alguna certificación de calidad como:

	Unidades	Porcentaje
a. iso	2	25,00
b. ieee	1	12,50
c. ninguno	4	50,00
d. otros	1	12,50



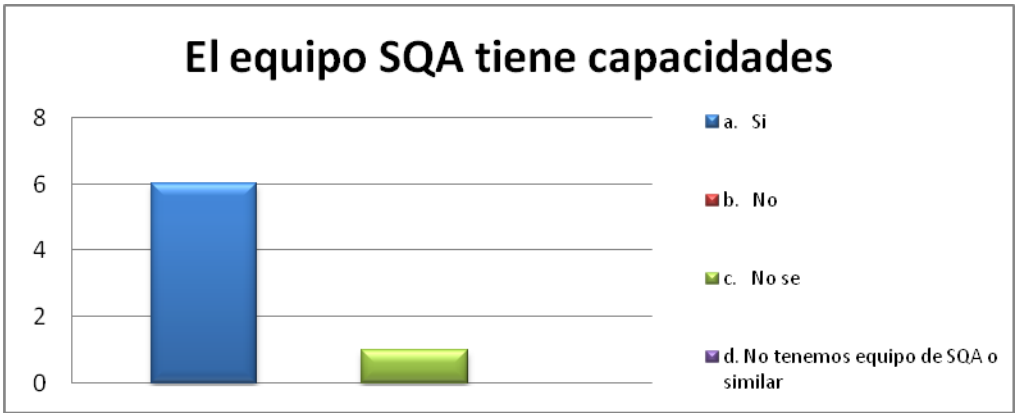
11. Luego de entregar el producto, el cliente (interno o externo), le pide realizar cambios a la aplicación.

	Unidades	Porcentaje
a. si	7	100
b. No		



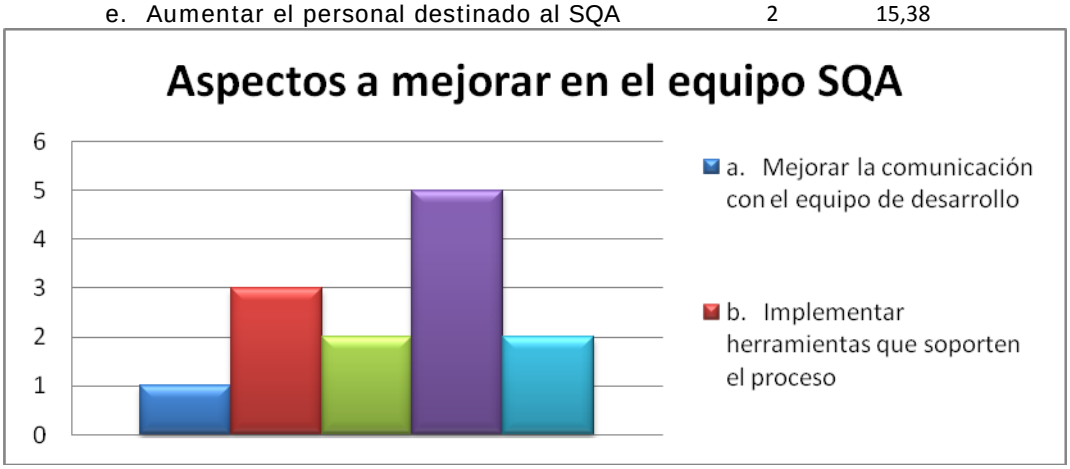
12. Considera al equipo SQA con capacidades suficientes para verificar el software entregado?

	Unidades	Porcentaje
a. Si	6	85,71
b. No		
c. No se	1	14,29
d. No tenemos equipo de SQA o similar		



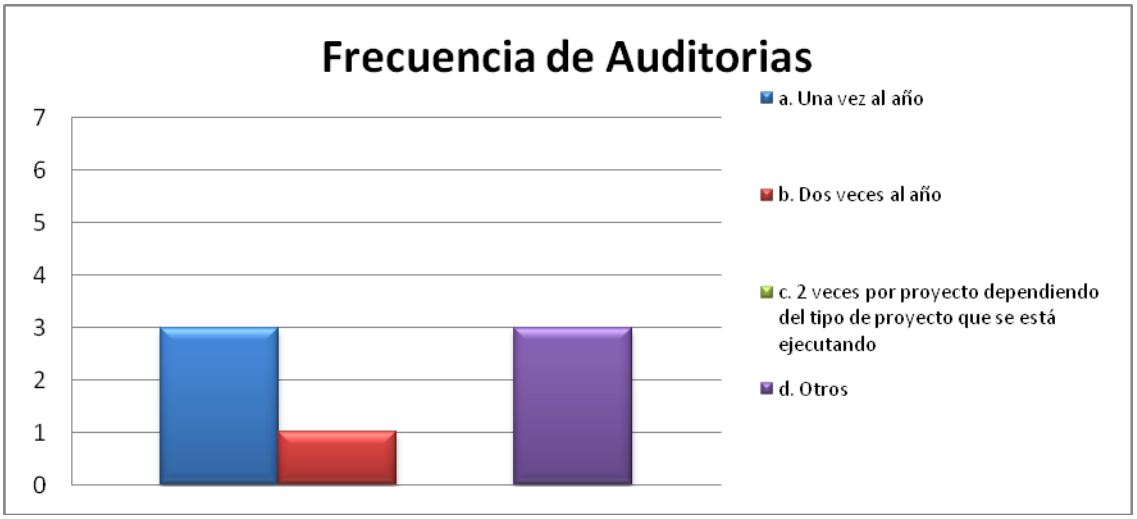
13. Qué aspectos cree usted que debería mejorar su equipo de SQA?

	Unidades	Porcentaje
a. Mejorar la comunicación con el equipo de desarrollo	1	7,69
b. Implementar herramientas que soporten el proceso	3	23,08
c. Mejorar la definición del proceso SQA	2	15,38
d. Formar personal SQA	5	38,46
e. Aumentar el personal destinado al SQA	2	15,38



14. Con qué frecuencia se ejecutan las auditorias al área de sistemas que desarrolla SW:

	Unidades	Porcentaje
a. Una vez al año	3	42,86
b. Dos veces al año	1	14,29
c. 2 veces por proyecto dependiendo del tipo de proyecto que se está ejecutando		
d. Otros	3	42,86



7.3.CONCLUSIONES

- En el medio ecuatoriano sí se tiene conocimiento sobre las normas de calidad en el desarrollo del software aunque aparentemente no sobre los beneficios que podrían obtener al estar certificados, puesto que pocas empresas disponen de este tipo de reconocimiento a la calidad.
    - Generalmente las empresas se privan de una certificación ya que los costos que la obtención de uno de estos reconocimientos implica generan un muy elevado costo que no siempre es tomado en cuenta dentro del presupuesto de dichas empresas.

- Las empresas estudiadas concentran sus actividades principalmente en el desarrollo de sistemas además tienen también interesante actividad en servicio y aplicaciones web.

- La mayoría de las empresas que se dedican a la actividad del, desarrollo de software lo realizan para uso externo, es decir el foco de los cliente de las empresas estudiadas son externos de tal manera que sería más importante garantizar la calidad de los productos y así ganar una reputación con los clientes

- En su totalidad las empresas participantes en este estudio coinciden que la calidad es muy importante dentro del proceso de desarrollo de software

- Los resultados revelan que el 100 % de las empresas objeto de este estudio, buscan la calidad durante todo el proceso de desarrollo de SOFTWARE

- Las empresas que desarrollan SOFTWARE, buscan un apoyo significativo en metodologías como COBIT, CMMI, RUP, MSF, ISO

- El estudio revela que las empresas tienen una experiencia no menor a un año utilizando las metodologías de apoyo a la calidad antes mencionadas.

- El estudio realizado muestra que uno de los principales requerimientos en el equipo de SQA es que no tienen facilidad en la formación del personal, es decir no existen en el mercado cursos o seminarios que ayuden al personal de SQA, a ampliar sus conocimientos

CAPITULO 8.- CONCLUSIONES Y RECOMENDACIONES

La responsabilidad de la calidad del Software, pasa por las manos de cada participante del proyecto y el no tomar las decisiones adecuadas en un momento determinado se pueden transformar en grandes problemas cuando el sistema llega a producción.

Algunas sugerencias para asegurar la calidad de forma continua son:

- Garantizar una comunicación eficiente entre los miembros del equipo y entre el equipo y el cliente;
- Preocuparse por la calidad desde el principio;
- Promover revisiones y pruebas cuanto antes sea posible;
- Evaluar constantemente la calidad del producto y del proceso;
- Acompañar a través de indicadores la evolución del proyecto y la calidad del producto resultado de cada actividad.

Se puede concluir también que considerar la calidad del software es tan importante que debe concretarse en entender inicialmente el concepto de “calidad del software”, crear el conjunto de actividades para su implementación, llevar a cabo tales actividades durante todo proyecto, utilizar métricas para desarrollar estrategias que mejoren el proceso del software y, como consecuencia, mejoren la calidad el producto final.
La Garantía de calidad del software Consiste en mantener un compromiso de mantener un determinado nivel de calidad en función de los objetivos involucrados en el proyecto de desarrollo.
Los puntos básicos que se debe tener en cuenta a la hora de construir Software bajo un Sistema de Garantía de Calidad, se refiere a la implementación de un plan empresarial basado en un procedimiento de calidad, es decir una estrategia completa por la cual una compañía entera se compromete a utilizar todos los recursos para satisfacer a sus beneficiarios en términos de calidad, costo y plazo. Se debe desarrollar un "espíritu de calidad" y todos los empleados deben compartirlo para que la gestión de calidad total tenga éxito.
Pero para que los empleados se encuentren alineados con el espíritu de calidad es muy importante culturizar a los recursos en materia de calidad y dotar al personal de Aseguramiento de calidad del conocimiento necesario para que pueda ser capaz de planificar y efectuar las pruebas del software que se desarrollen.
En este documento se ha incluido un conjunto de conceptos y lineamientos que servirán al personal como una guía informativa básica para que pueda desempeñar su trabajo de una forma más profesional.
En cuanto a las normas y principios de calidad del software se puede decir que la Certificación o acreditación es el reconocimiento escrito por parte de un tercero independiente de que un servicio, producto o sistema cumple con cierto nivel de calidad. La certificación generalmente se basa en una norma (preferentemente internacional). Para esto es importante que las empresas conozcan sobre las normas y principios en los que se basa la calidad del software, para las pequeñas y medianas empresas de software se recomienda la implantación de sistemas de la calidad basados en las normas ISO 9000 ya que puede constituir el mejor camino a seguir pues estas normas definen los requisitos básicos de la

organización y por otro lado la certificación le confiere un prestigio importante ante sus clientes.

Sin embargo muchas empresas anuncian a bombo y platillo que implementan una metodología y logran obtener una certificación, en las ofertas del negocio este constituye un punto de decisión de compra muy importante y referente para los clientes.
Pero en muchas ocasiones estos procesos certificados no se aplican, debidos principalmente al poco tiempo con el que se cuenta para desarrollar el proyecto, porque no se dedican y porque las empresas normalmente no se preocupan por tener personal formado o dedicado a ello los recursos ni los medios necesarios para el desempeño y cumplimiento de la metodología, y porque las empresas normalmente no se preocupan por tener personal formado o dedicado a ello.

Una vez alcanzado cierto nivel de manejo de la metodología de pruebas es posible también utilizar herramientas que sirvan de apoyo a las pruebas de software, las herramientas son aplicaciones que existen en el mercado de la tecnología y que por medio de su utilización se puede optimizar tiempo y recursos, centrando un mayor esfuerzo en los sectores críticos del Software

Una diferencia interesante entre las Pruebas y otras disciplinas del proceso de desarrollo de Software es que esencialmente es una tarea que encuentra y pone de manifiesto las debilidades del producto de software. Existen cuatro elementos que son relevantes al momento de definir las pruebas: Confiabilidad, Costo, Tiempo y Calidad.
En la medida que se deseen pruebas confiables y un software de calidad, el tiempo y el costo se incrementarán.

Los tipos de Pruebas que se ejecutan en el desarrollo de un plan de pruebas son:

- Pruebas de Validación su objetivo es demostrar que el software desarrollado es el que el cliente quiere, que satisface sus requerimientos.

- Pruebas de defectos su objetivo es revelar inconsistencias entre el programa y su especificación.

Una diferencia interesante entre las Pruebas y otras disciplinas del proceso de desarrollo de Software es que esencialmente es una tarea que encuentra y pone de manifiesto las debilidades del producto de software. Existen cuatro elementos que son relevantes al momento de definir las pruebas: Confiabilidad, Costo, Tiempo y Calidad.
En la medida que se deseen pruebas confiables y un software de calidad, el tiempo y el costo se incrementarán.

La Calidad del Software es definitivamente importante ya que uno de los enfoques fundamentales es el cliente, por consiguiente, el propósito de calidad es proporcionarle al cliente una oferta apropiada con procesos controlados y al mismo tiempo garantizar que esta mejora no se traduzca en costos adicionales. Es posible mejorar un gran número de problemas a un bajo costo. Sin embargo, cuanto más cerca se está de la perfección, más se elevan los costos.

En lo absoluto, para las compañías del sector privado en realidad no es una cuestión de satisfacer exhaustivamente las expectativas del cliente ("sin defectos"), sino de satisfacerlas mejor que la competencia.

Lo opuesto a la calidad, (o un **defecto de calidad**), también tiene un costo. De hecho, generalmente es más costoso corregir defectos o errores que "hacerlo bien" desde el comienzo. Además, el costo de un defecto de calidad es mayor cuanto más tarde se detecta. Ya que el impacto que causa un defecto detectado durante la producción por un cliente generaría insatisfacción del cliente, procesamiento del incidente, control del cliente, costos de envío, etc.

La única manera de realizar una mejora en la calidad de los productos que se desarrollan es verificar en que estado nos encontramos a través de la medición, evaluación y la aplicación de técnicas, métodos y herramientas asociadas con el proceso de desarrollo es importante también establecer una diferencia entre las técnicas de verificación y validación

- La verificación consiste en determinar si la aplicación que estamos construyendo se encuentra libre de defectos
- La Validación consiste en determinar si la aplicación que se esta desarrollando cumple a cabalidad con las necesidades del usuario

La recomendación para una adecuada ejecución de un sistema de control de calidad del Software es que:

- Cada empresa informática debería contar con profesionales en calidad y aportar su experiencia en todos los proyectos que se ejecuten, si no cuentan con personal con este perfil capacitarlos, formarlos para que vayan creciendo con el proceso.
- Aplicar adecuadamente las metodologías, y tener un seguimiento y control exhaustivo con los clientes.
- Realizar periódicamente auditorías externas de calidad y de metodología, si es posible que sean oficiales es mucho mejor, como si fueran inspecciones.

Cumpliendo con estas recomendaciones se aseguraría la calidad de la empresa y de sus profesionales.

El número de casos de prueba necesarios para probar un programa software es infinito, por lo que es imposible conseguir un programa totalmente probado, siendo además el proceso de prueba muy costoso. Por estos motivos las técnicas para la generación de casos de prueba tratan de encontrar de forma eficiente un conjunto pequeño de casos de prueba que permitan cumplir un determinado criterio de suficiencia.

El proceso de pruebas no permite garantizar que un software se realiza de forma correcta, únicamente permite garantizar que no se entregará el sistema si este no responde con éxito a determinados tipos de pruebas.

GLOSARIO DE TÉRMINOS

- Análisis: Fase en la que se definen las razones y justificaciones de los sistemas de información.
- Análisis de Requerimientos: Define el momento del "qué de los sistemas de información".
- Aplicación: Aplicación sobre la cual el software va dirigido.
- Aseguramiento de Calidad (SQA): Determina si las necesidades de los usuarios están siendo satisfechas adecuadamente.
- Auditoría Informática: Proceso de recoger, agrupar y evaluar evidencias para determinar si un sistema informatizado salvaguarda los activos, manteniendo la integridad de los datos. Lleva a cabo eficazmente los fines de la organización, administra eficientemente los recursos.
- Codificación: Conversión del diseño de sistemas de información a instrucciones ejecutables.
- Calidad en el Software: Totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas.
- Control de la Calidad: Técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad.
- Certificación de la Calidad: Validez, que demuestra que la organización es capaz de desarrollar productos y servicios de calidad, bajo estándares certificados.
- CMM: Capability Maturity Model.
- Corrección: Grado en que un programa al corregirse satisface las especificaciones y consigue los objetivos de la misión encomendada por el usuario final.
- Control Interno: Actividades operativas claves destinadas a prevenir los riesgos efectivos y potenciales a los que se enfrentan las organizaciones.
- Controles Preventivos: Controles que evitan el hecho, como un software de seguridad que impida los accesos no autorizados al sistema.
- Controles Detectivos: Controles que manifiestan cuando fallan los controles preventivos para tratar de conocer cuanto antes el evento.
- Controles Correctivos: Controles que facilitan la vuelta a la normalidad cuando se han producido incidencias.
- Datos: Grupo de elementos que tienen forma y contenido similares en estructura de implantación y en la composición.
- Documentación: Documentos, principales para establecer e implantar un sistema de calidad; puede haber manuales a nivel empresa, departamento, producto especifico.
- Desarrollo Iterativo: Define a cada una de las fases del proceso de desarrollo de una aplicación, requerimientos, análisis, diseño, implementación, pruebas, evaluación; es repetida y refinada hasta que finalmente se cumplen los requerimientos del sistema.
- Diseño: Cambio que sufren los sistemas de información a representaciones, como tablas, gráficas, basadas en lenguajes y estructuras de datos.

- Eficiencia: Cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.
- Experiencia del Usuario: La familiaridad de los usuarios con computadoras o software similares.
- Ejecución: Optimización del uso de hardware y software al implementar los productos de software.
- Factores de Calidad: Constituyen los bloques fundamentales de construcción sobre los que se edifica la calidad. Estos factores son el medio por el cual se traduce el término de calidad al lenguaje de las personas que manejan la tecnología.
- Fiabilidad: Grado en que se puede esperar que una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.
- Flexibilidad: Esfuerzo requerido para modificar una aplicación en funcionamiento.
- Gestión de la Calidad: Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implantan por medios tales la planificación, el control, el aseguramiento y la mejora de la calidad en el marco del sistema de calidad.
- ISO 9000: Normativa de calidad en la gestión y aseguramiento de calidad de software. Define los conceptos y directrices de la calidad de software.
- Lenguaje de Programación: Lenguaje o paquete computacional seleccionado para desarrollar el software.
- Madurez del Desarrollador: Experiencia en el desarrollo de software similar.
- Metodología de Desarrollo: Uso de algún método establecido para la creación de software.
- Métricas de Software: Aquella aplicación continúa de técnicas basadas en la medida de los procesos de desarrollo del software, para producir una información de gestión significativa.
- Métricas de Calidad: Métricas, que se argumentan que éstas deben ser enunciadas y utilizadas para administrar el proceso de desarrollo y debe ser conforme al producto de software particular.
- Mejores Prácticas: Conjunto de acciones cuya principal disciplina en los equipos de desarrollo de software es garantizar la calidad mediante la reducción de fallas al liberar el sistema.
- Niveles de Madurez: procesos de desarrollo de software en una escala de cierta cantidad de niveles en donde se tiene en cuenta aspectos muy variados de los procesos de desarrollo, como el grado de ambigüedad de las especificaciones, la verificación independiente de la fiabilidad de los programas, etc.
- Operaciones del Producto: Define las características operativas del software.
- Planeación: Estimación de recursos, definición de responsabilidades y actividades así como su secuencia de ejecución.
- Prueba: Procedimientos para verificar que no existan errores.
- Portabilidad: Esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.
- Producto de Software: Todo aquel producto final que se obtiene como resultado de la aplicación de los procesos de la ingeniería de software.

- Reusabilidad: Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.
- Riesgos: elementos que se presenten en cualquier situación y que estos puede implicar que el software falle.
- Sistemas de Información: Son sistemas que se desarrollan con diferentes propósitos para procesamientos de datos, administración para apoyo en la toma de decisiones.
- Sistema de Calidad: Representación de la estructura organizativa, procedimientos, procesos y recursos necesarios para implantar la gestión de calidad, adecuándose a los objetivos de calidad de la empresa.
- SEI: Software Engineering Institute.
- UML: Unified Modeling Language. Es el diseño de aplicaciones basada en componentes.

BIBLIOGRAFÍA

1. [AZAÑEDO. A] "El Valor de las Pruebas", Alberto de Cae Azañedo

2. [BARREIRO, E] "La calidad del software, Enrique Barreiro

3. [BEDINI. A] "Calidad Tradicional y de Software Alejandro Bedini

4. [BORJA.J] "La Calidad, según Borland, Jordi Borja

5. [BOEHM. B] "Barrh Boehm, Software Engineering Economics, Prentice-Hall 1981"

6. [CALDERON. J.] "Curso de auditoría y garantia de calidad, Jorge López Calderón

7. [CARRASCO, O] "Un enfoque actual sobre la calidad del software Oscar Fernández Carrasco, Delba García León Alfa Beltrán Benavides.

8. [CASALLAS, R] "Mejoramiento del Proceso de Pruebas en un Contexto de desarrollo de Software Globalizado",Rubby Casallas, Darío Correal, Nicolás López "

9. [CORDOBA, J] "Modelo de Calidad para Portales Bancarios" Julio Córdoba, Cristina Cachero

10. [FERNANDEZ. O] "Un enfoque actual sobre la calidad del software", Oscar M. Fernández Carrasco,

11. [FERRERO.C] "Guia-evaluacion-Software" Carlos Ferrero Presidente del Consejo de Ministros

12. [GARCES.V] "Apuntes De Clase Ingeniería Del Software Ii (Uisek)" Ing. Vicente Garcés

13. [GARCIA. J] "Articulo sobre el CMM-CMMI", Joaquín Gracia

14. [GONZALEZ. J] "Hacia la Medición de Calidad en Uso Web" Julia González Rodríguez , Luis Olsina

15. [GRIMAN. A PEREZ. M] "Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales" Anna. C Grimán, María Pérez, Luis. E Mendoza

16. [GUZMAN.O] "Aplicación práctica del diseño de pruebas de software a nivel de programaciónGuzmán Cortés

17. [HERNAM. R] "Calidad de los proyectos informáticos" Rafael Hernam Pérez

18. [HORCH. J] Practical Guide to Software Quality Management, Second Edition. by John W. Horch

19. [ISHIKAWA, KAORU.] "¿Qué es Control Total de Calidad?" Ishikawa, Kaoru. La modalidad Japonesa. Editorial Norma.

20. [ISO 9001:2000] ISO/IEC; Software engineering — Guidelines for the application of ISO 9001:2000 to computer software; ISO/IEC 90003.

21. [ITEC] "Calidad de Software – ITEC "Leonardo DaVinci" – 2006"

22. [LAWRENCE.E] "A Software Quality Model and Metrics" Lawrence E. Hyatt and Linda H. Rosenberg, Ph.D.

23. [LEON. L] "Caracterización de la Prueba de Software" Luis Vinicio León Carrillo

24. [LOPEZ. A] "Solución de IBM Rational para mejorar la calidad de aplicaciones software" Ana López-Mancisidor Rueda

25. [MEDINA, M] "¿Está usted controlando la calidad de su software?", Ing. Maricel Medina Mora

26. [MELES. J] "Calidad CMMI UTN", Ing. Judith Meles

27. [MENDEZ. E; PEREZ. M] "Actas de Talleres de Ingeniería del Software y Bases de Datos", Edumilis Méndez, María Pérez, Luis E. Mendoza,

28. [MENDOZA.L] "SISTEMAS DE INFORMACIÓN III TEORÍA" Prof. Luis Eduardo Mendoza M

29. [MIRANDA.L] "La calidad, su evolución histórica y algunos conceptos y términos asociados", Lic. Sandor Luis Miranda, Prof. Dr. Ing. Arturo Luis Romero

30. [MONTILVA. J] "Modelos de procesos para el desarrollo de software orientado a objetos", Jonás Montilva;

31. [PERRY, WILLIAM E.] "Quality assurance for information systems". QED technical publishing group, PERRY, WILLIAM E.

32. [POLO. M] "Mantenimiento Avanzado de Sistemas de Información Pruebas del Software "Dr. Macario Polo Usaola

33. [PRESSMAN R] Ingeniería del Software, Un enfoque practico. Tercera edición. McGraw Hill PRESSMAN, Roger S

34. [RAGHU. S] International Standard,Iso/Iec 12207,Software Life Cycle Processes/Federal Aviation Administration/Washington,DC,USA, Singh RAGHU,

35. [RAJA.E] "Casi todas las pruebas del software" Elena Raja Prado

36. [ROCA. M.Jose] "Pruebas de Integración de Productos: Un enfoque práctico", María José Roca V.

37. [ROCHE. D] "Aseguramiento de Calidad en el Software" Daniel Roche

38. [RODRIGUES. P] "Sistema de calidad del SOFTWARE en Metro de Madrid (Pruebas)", Paz Rodrigues,Antonio Fuentes

39. [RODRIGUEZ. A] M.Sc. Abilio Marrero Rodríguez

40. [RODRIGUEZ. L] "Conflictos actuales en el desarrollo de software por terceros: El problema de la calidad " Software lifecycle Optimization. Luis Rodríguez Berzosa

41. [RUALES. G] "Lineamientos Metodologicos Para La Elaboración Del Plan De Tesis" Ing. Gonzalo Ruales

42. [SENTLE.A] Conceptos basicos de calidad Norma ISO 9001:2000 Auditorias Internas de calidad Nuevo Loredo 2008-2010 Andres Sentle

43. [SIMMONS, RONALD A] "Software quality assurance (SQA) early in the acquisition process", SIMMONS, RONALD A.

44. [SOMMERVILLE. I] Ingenieria de Software Septima edicion Ian Sommerville Pearson educación 2005

45. [STRAUB. P] "El costo de la calidad y el costo de la mala calidad" Pablo Straub AgileShift

46. [VALICA. E] "Software Testing Achievements – Dreams" Ecaterina Valică

47. [VEGA. C] "Mejores prácticas para el Establecimiento y aseguramiento de La calidad de software" "Vega Lebrún Carlos Rivera Prieto Laura Susana García Santillán Arturo"

48. [VILLA. M] Modelos de Evaluación y Mejora de Procesos: Análisis Comparativo - Manuel de la Villa, Mercedes Ruiz e Isabel Ramos

49. [VOS.T] Testeo Testeo de de Usabilidad. Tanja Vos

ANEXOS

Anexo001 PLANTILLA PARA DOCUMENTAR UN CASO DE PRUEBA.

CASO DE PRUEBA			
Nombre de la aplicación - Módulo:			
SubMódulo			
Transacción:			
Caso de uso:			
Fecha de la Prueba:			
Responsable:			
Escenario:			
Simulación y validación Entradas al sistema	**Estado**	**Observación**	**Registro**
Procesos que ejecuta en este caso de prueba			
Salidas Esperada			
Otras Validaciones (Caminos alternos)			
Compatibilidad (si el sistema interactúa con otros sistemas)			

Anexo002 LISTA DE CHEQUEO ASEGURAMIENTO DE CALIDAD

LISTA DE CHEQUEO ASEGURAMIENTO DE CALIDAD

Nombre de la aplicación

Fecha de la Prueba:

Responsable:

Registro de la Información

	Actividad	Estado		
		OK	K.O	NA
1	¿Existe alguien en su organización responsable por los procesos de pruebas?			
2	¿Tiene y usa un estándar para el plan de pruebas?			
3	¿Tiene y usa un estándar para las pruebas de unidad?			
4	¿Tiene y usa un estándar para el reporte de la ejecución de las pruebas?			
5	¿La planeación y ejecución de pruebas se realiza en paralelo con el proceso de desarrollo de software?			
6	¿Se verifica que las especificaciones estén correctamente implementadas?			
7	¿Se verifica que las expectativas del cliente sean satisfechas?			
8	¿Los probadores verifican la precisión y completitud de productos internos tales como el documento de requerimientos o los diseños?			
9	¿Los probadores reportan los defectos al equipo de desarrollo de software para corrección?			

10	¿Los probadores identifican la prioridad de los riesgos del negocio para el desarrollo del plan de pruebas?			
11	¿Existen objetivos de pruebas medibles para cada sistema de software que está siendo probado?			
12	¿Los objetivos están alineados con los riesgos del negocio?			
13	¿Se usan métricas para mejorar el proceso de aseguramiento de la calidad?			
14	¿Los probadores han definido pronósticos de defectos basándose en datos y experiencias previos?			
15	¿Existe un proceso de mejoramiento continuo para su proceso de pruebas?			
16	¿Los tipos de defectos están identificados?			
17	¿Se registra, acumula y se usan los datos de fallas para evaluar la efectividad del proceso de pruebas y producir un software libre de defectos?			
18	¿Se usan métricas para planear y evaluar el proceso de pruebas?			
19	¿Tiene un proceso de entrenamiento de probadores?			
20	¿El uso de una herramienta automatizada de pruebas es parte significativa de su proceso?			

Anexo003 LISTA DE CHEQUEO DE ESTÁNDARES DE PRESENTACIÓN Y FUNCIONALIDAD DE LA APLICACIÓN PARA FORMAS

Lista de chequeo de estándares de presentación y funcionalidad de la aplicación para formas

Nombre de la aplicación

Fecha de la Prueba:

Responsable:

Registro de la Información

	Actividad	Estado		
		OK	K.O	NA
1	¿Están claramente definidos los bloques de información (Frames)?			
2	¿Tiene los encabezados de título y nombre de aplicación correctos?			
3	¿Las etiquetas de los campos son claras y representativas?			
4	¿Tiene los colores estándar?			
5	¿Los campos fecha tienen el formato DD-MON-RRRR y se puede ingresar los datos como Ej: 12ago2001?			
6	¿La forma tiene la dimensión correcta?			
7	¿Los Radio Groups tienen un frame que los abarca?			
8	¿Los campos están alineados en forma correcta?			
9	¿Los campos requieren y tienen Tooltip?			
10	¿Están habilitados los botones del toolbar de manera adecuada y corresponden con las teclas de función?			

11	¿La forma realiza la función que se necesita?			
12	¿Es rápido y fácil el manejo de la forma?			
13	¿El tiempo de respuesta es adecuado?			
14	¿El orden de navegación de los campos es el correcto?			
15	¿Los mensajes graves son manejados adecuadamente?			
16	¿Si el reporte requiere mucho tiempo, esto le es notificado al usuario?			
17	¿Está la forma documentada?			
	Forma			
18	¿Tiene la forma la descripción y su título de acuerdo con los estándares?			
	Título del frame			
19	¿Está el título en mayúscula inicial?			
20	Si el frame es de un solo registro, ¿está su título en singular?			
21	Si el frame es multi registro, ¿está su título en plural?			
22	¿Está localizado el título en la parte superior izquierda del frame?			
	Campos			
23	¿Tiene el contenido del campo la alineación adecuada, de acuerdo con su tipo de dato?			
24	Si existen varios campos organizados verticalmente, ¿están alineados todos a la izquierda?			
	Etiquetas			

25	Si la organización NO es tabular, ¿están situadas las etiquetas a la izquierda del campo al que pertenecen?			
26	Si la organización SÍ es tabular, ¿están situadas las etiquetas en la parte superior del campo al que pertenecen?			
27	Si la organización SÍ es tabular, ¿están las etiquetas centradas?			
28	¿Están las etiquetas en mayúscula inicial?			
29	¿Están las etiquetas sin los dos puntos al final?			
30	Si existen varios campos organizados verticalmente, ¿están alineadas todas las etiquetas a la derecha?			
31	¿Están las etiquetas formadas de manera que no utilicen abreviaturas ni expresiones de solicitud?			
32	Si están ubicados horizontalmente, ¿están en la parte inferior de la pantalla?			
33	Si están ubicados verticalmente, ¿están a la derecha de la pantalla?			
34	Los botones organizados horizontalmente, ¿tienen la misma altura?			
35	Los botones organizados verticalmente, ¿tienen el mismo ancho?			
36	¿Está colocada la opción más frecuente a la izquierda o en el tope, según corresponda?			

Anexo004 LISTA DE CHEQUEO DE ESTÁNDARES DE PROGRAMACIÓN O CÓDIGO

Lista de chequeo de estándares de programación o código

Nombre de la aplicación

Fecha de la Prueba:

Responsable:

Registro de la Información

	Actividad	Estado		
		OK	K.O	NA
1	¿Está el código indentado a, por lo menos dos espacios?			
2	¿Están ordenados alfabéticamente las constantes, variables y cursores?			
3	¿Están alineados a la izquierda las constantes, variables y cursores?			
4	¿Están alineados a la izquierda la definición del tipo de dato de las constantes, variables y cursores?			
5	¿Está definida sola una constante, variable o cursor por línea?			
	Documentación			
6	¿Está toda la documentación en una línea diferente al código que se está documentando?			

7	¿Comprende la documentación de funciones/ procedimiento tres partes: una descripción general de lo que hace la función o procedimiento, la descripción de los parámetros de entrada y la descripción de los posibles valores y/o parámetros de salida?			
	Parámetros			
8	¿El nombre de los parámetros empieza con la letra *p* minúscula y es significativo?			
	Constantes			
9	¿El nombre de las constantes empieza con la letra *c* minúscula y es significativo?			
	Variables			
10	¿El nombre de las variables empieza con la letra *v* minúscula y es significativo?			
	Instrucciones Select, Insert, Update y Delete			
13	¿Están todas las instrucciones Select, Insert, Update y Delete escritas en minúsculas, a excepción de variables que hagan referencia a campos de las formas?			
	Instrucciones Select			
14	¿Están las cláusulas Select, Into, From, Where, Order BY, Group BY y Having escritas en líneas diferentes?			
	Instrucciones Insert			
15	¿Están las cláusulas Insert Into y Values escritas en líneas diferentes?			
	Instrucciones Update			

16	¿Están las cláusulas Update, SET y Where escritas en líneas diferentes?			
17	¿Está cada columna que se actualice en una línea diferente?			
18	¿Están todas las columnas que se actualicen alineadas a la izquierda?			
	Instrucciones Delete			
19	¿Están las cláusulas Delete y Where escritas en líneas diferentes?			

Anexo005 ENCUESTA EJECUTADA A LAS EMPRESAS DE DESARROLLO DE SOFTWARE

Estado de la calidad en una empresa de desarrollo de Software

Objetivo de la Encuesta*: Adquirir una visión general de las empresas que desarrollan SOFTWARE para uso local o para uso comercial y evaluar el conocimiento de estas empresas para asegurar la calidad del SOFTWARE.*

Nombre de la Organización: _____
Tipo de desarrollo que se dedica _____
Nombre de la Persona Encuestada: _____
Cargo/Función: _____
Tiempo de experiencia: _____ Fecha: _____

A continuación se listara una serie de preguntas, por favor marque con una 'X' en los espacios propuestos (dentro de los paréntesis), la respuesta que se adapte a su realidad, de favor le pedimos también nos ayude contestando con las preguntas abiertas, de antemano mil gracias por su colaboración.

1. El software que desarrollan consiste de:
   a. ( )Sistemas;
   b. ( )Servicios,
   c. ( )App.web
   d. ( )Prototipos;
   e. ( )Otros: _____

2. El software que desarrollan es para uso:
   a. ( )Interno
   b. ( )Externo
   c. ( )Otros:_____

3. Qué papel desempeña la calidad en el software que se desarrolla, categorizando el nivel de calidad que se requiere, se puede decir que la calidad :

    a. ( )Si es Importante
    b. ( )Es algo importante
    c. ( )No es nada importante

4. En qué fase del ciclo de vida del software se busca la calidad del software:
    a. ( )Durante el análisis
    b. ( )Durante el diseño
    c. ( )Durante la planificación
    d. ( )Durante la Implementación
    e. ( )Durante la etapa de pruebas
    f. ( )Todo el ciclo de vida

5. Que metodología, procedimientos o lineamientos metodológicos de apoyo en la calidad software, utiliza su organización para el desarrollo de software
    a. ( )Modelado de Objetos (UML)
    b. ( )Gestión de proyectos de software
    c. ( )Metodologías de Apoyo en la calidad de software ….. (COBIT, CMMI, RUP, MSF,ISO)
    d. ( )Otros:_____

6. Mencione, los puntos claves en los que consiste el desarrollo de SOFTWARE con el uso de dicho procedimiento, metodología, lineamiento
   (Ref: pregunta 5.):

_____

_____

_____

_____

7. Qué tiempo ha utilizado dicho procedimiento, metodología… para el desarrollo de software en su organización :
   a. ( )Menos 1 año
   b. ( )Entre 1-2 años
   c. ( )Mas de 3 años

8. Considera importante utilizar una metodología de desarrollo de software, que garantice, los mínimos estándares de calidad :
   a. ( )Muy importante
   b. ( )Importante
   c. ( )Nada Importante

9. El departamento o área de sistemas que desarrolla software, cuenta con alguna certificación de calidad como:
   a. ( )ISO
   b. ( )IEEE
   c. ( )Ninguno
   d. ( )Otros:_____

10. Por favor mencione de qué manera garantiza su organización la calidad en los sistemas que desarrolla.

_____

_____

_____

_____

11. Luego de entregar el producto, el cliente (interno o externo), le pide realizar cambios a la aplicación.
   a. ( ) Si.
      a.1. De qué manera maneja su organización los cambios Solicitados?

      _____
      _____
      _____
      _____
      _____
      _____
   b. ( )No

12. Considera al equipo SQA con capacidades suficientes para verificar el software entregado?
    a. ( )Si
    b. ( )No
    c. ( )No se
    d. ( )No tenemos equipo de SQA o similar

13. Qué aspectos cree usted que debería mejorar su equipo de SQA?
    a. ( )Mejorar la comunicación con el equipo de desarrollo
    b. ( )Implementar herramientas que soporten el proceso
    c. ( )Mejorar la definición del proceso SQA
    d. ( )Formar personal SQA
    e. ( )Aumentar el personal destinado al SQA

14. Con qué frecuencia se ejecutan las auditorias al área de sistemas que desarrolla software:
    a. ( )Una vez al año
    b. ( )Dos veces al año
    c. ( )2 veces por proyecto dependiendo del tipo de proyecto que se está ejecutando
    d. ( )Otros_____

Muchas gracias por su colaboración y por su tiempo…