UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN

"ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE"

CALIDAD EN LAS ORGANIZACIONES (EMPRESAS E INSTITUCIONES)

TRABAJO DE SEMINARIO

QUE PARA OBTENER EL TITULO DE

INGNIERO MECANICO ELECTRICISTA

P R E S E N T A:

LUIS MIGUEL FRANCHINI GÓMEZ

ASESOR:
ING. EMILIANO FONES ESPINOZA
CUAUTITLAN IZCALLI, EDO. DE MEX

2002

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
DEPARTAMENTO DE EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN. Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 51 del Reglamento de Exámenes Profesionales de la FES-Cuautitlán, nos permitimos comunicar a usted que revisamos el Trabajo de Seminario
Calidad en las organizaciones (empresas e instituciones).
"Aseguramiento de la calidad del software"

que presenta el pasante. Franchini Gómez Luis Miguel
con número de cuenta: 8932775-5 para obtener el título de
Ingeniero Mecánico Electricista

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXÁMEN PROFESIONAL correspondiente, otorgamos nuestro VISTO BUENO

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 27 de septiembre de 2002

MODULO	PROFESOR	FIRMA
I	Ing. José Juan Contreras Espinosa	
II	Ing. Emiliano Pones Espinoza	
IV	Ing. José Luz Hernández Castillo	

Agradecimientos

Agradezco a mis padres porque por ellos soy en gran parte como soy.

Agradezco a mi Mamá María del Socorro porque siempre me ha enseñado a luchar y me ha enseñado que las cosas se debe hacer bien sin importar el esfuerzo que implique y por su gran amor.

Agradezco a mi Papá Luis Humberto porque siempre me ha apoyado y siempre me ha enseñado a conducirme con honestidad.

Agradezco a mis hermanos Carlos Humberto y Jorge Alberto por todo su apoyo y comprensión.

Agradezco a mi Mamá Meche por todo su amor y por todas las cosas que me enseñó desde niño y que me han servido durante toda mi vida y que me servirán siempre.

Agradezco a mi tía Paty por todo su apoyo y por todas las cosas que me enseñó que también me han servido mucho.

Agradezco a mi novia Miriam por todo su apoyo y comprensión y por ser el amor de mi vida.

Agradezco a la Facultad por la gran oportunidad que me ha dado de ser alguien en la vida y por abrirme siempre las puertas.

Agradezco a los profesores por todas sus enseñanzas y dedicación.

Agradezco a mi asesor el Ingeniero Fones por todo su apoyo y paciencia para lograr la terminación de este trabajo.

Introducción.

Conceptos y definiciones

El aseguramiento del software comprende una serie de actividades sistemáticas y planeadas que nos aseguran que los procesos del mismo y que los productos cumplan con: los requisitos, los estándares, y los procedimientos.
Los "procesos" incluyen todas las actividades implicadas en el diseño, desarrollo, actualización, y mantenimiento del software; los "productos" incluyen el software, los datos asociados, su documentación, y todo el papeleo de soporte y reportería.

Hay tres actividades implicadas en el ciclo de vida del software, dichas actividades que interactúan entre ellas son: administración del software, ingeniería del software, y aseguramiento del mismo. La administración del software comprende una serie de actividades implicadas en la planeación, control, y dirección del proyecto. La ingeniería del software abarca una serie actividades que analizan requisitos, desarrollan diseños, escriben código, y estructuran bases de datos. El aseguramiento del software se cerciora de que los esfuerzos de la administración y de la ingeniería se unan para dar como resultado un producto que resuelva todos sus requisitos.

Este trabajo se debe leer como propuesta de las actividades que pueden ser vitales para el éxito de un proyecto de software, algunas consideraciones para que una organización aumente su probabilidad de éxito se dan en el capítulo 1 llamado "Estableciendo las actividades para el aseguramiento de un proyecto de software".

Metas del aseguramiento del software

El desarrollo del software, como cualquier actividad compleja de desarrollo, es un proceso lleno de riesgos, los cuales pueden ser de dos tipos: técnicos y de planeación; es decir, los riesgos que

ocasionan que el software no realice lo previsto, o sea demasiado difícil de operar, modificar, o mantener, son riesgos técnicos, mientras que los riesgos que hacen que el proyecto supere el costo o el tiempo de desarrollo estimado, son riesgos de planeación. La meta del aseguramiento del software es reducirlos en lo posible. Por ejemplo, se deben fijar estándares de.codificación con el fin de establecer una calidad mínima del código; si no se hace, existe el riesgo de que el código requiera un retrabajo; si se fijan, pero no hay un proceso explícito de difusión de dichos estándares existe un alto riesgo de que el código producido por algunos programadores sea de baja calidad. El incluir un proceso de aseguramiento de la calidad del software disminuye el riesgo de que el producto sea inaceptable.

De la misma forma, no contar con un sistema de reporte de no conformidades y acciones correctivas aumenta el riesgo de que los problemas en el software sean olvidados y no resueltos, o que los problemas importantes no tengan la atención prioritaria que ameritan. Otros ejemplos relacionados con el riesgo se pueden proporcionar para apoyar todas las actividades en este trabajo. El objetivo de estas actividades es ayudar a reducir estos riesgos asegurando así la calidad del software

Propósito de este trabajo

El propósito de esta guía de aseguramiento del software es proporcionar asistencia, en la forma de administrar y liderar un proyecto, para los encargados y responsables de la adquisición y del desarrollo de software y para establecer requisitos del aseguramiento del mismo.

El presente trabajo pretende ser un tutorial más que un material didáctico. Se espera que la persona que lo lea, encuentre en él una guía útil para formular y tratar las necesidades de un proyecto de software relacionadas con el aseguramiento de la calidad.

En él se intentará sugerir y dar pauta a actividades que garanticen la calidad del producto la cual se puede lograr con: el aseguramiento de la calidad del software, la ingeniería de calidad del software,

verificación y validación del mismo, reporte de no conformidades y su acción correctiva, y seguridad del software. En el capítulo 1 se establecen las actividades para el aseguramiento de un proyecto de software, y esta pensado para analizar y asistir a los administradores en el comienzo de una nueva actividad en el aseguramiento o en la mejora un programa existente.

Glosario de términos

Calidad – La calidad puede ser un concepto confuso, en parte porque las personas visualizan la calidad con relación a diferentes criterios según su papel individual en la cadena de producción y de comercialización. Además el significado de calidad ha venido evolucionando conforme a la profesión de la calidad ha crecido y madurado, algunas definiciones de calidad son:

- Perfección
- Consistencia
- Eliminación de desperdicio
- Rapidez de entrega
- Cumplimiento de políticas y procedimientos
- Proporcionar un producto bueno y utilizable
- Hacerlo bien a la primera
- Agradar o satisfacer a los clientes
- Servicio total al cliente[1]

Certificación de calidad – Es el proceso mediante el cual un organismo o institución competente otorga a una empresa un reconocimiento de calidad de algún producto, proceso de producción o metodología de trabajo el cual sirve a dicha compañía para demostrar en el mercado su calidad y competitividad.

Ingeniería de software - requisitos de un sistema de gestión de calidad de carácter general que cubre el desarrollo de cualquier producto (ISO 9001) y ha publicado directrices específicas para aplicar esa norma al desarrollo de software (ISO 9000-3). Una organización que ponga en práctica un sistema de gestión de calidad según esa norma puede ser auditada y recibir una certificación formal de su proceso de desarrollo.

Los ingenieros informáticos están implicados en un gran número de áreas de aplicación, que cada vez son más. Algunos ejemplos son la realización de transacciones rápidas de valores en el mercado bursátil, el almacenamiento, intercambio y presentación de información en

[1] Administración y control de la calidad. James R. Evans

Internet, los videojuegos, la mejora de imágenes obtenidas por telescopios y el control de marcapasos cardiacos. En todos los casos, los principios de la ingeniería de software ayudan a garantizar que los sistemas resultantes son fiables y funcionan del modo requerido.

ISO - Es el organismo internacional de normalización (International Standards Organization), creado en 1947 y que cuenta con 110 estados miembros representados por sus organismos nacionales de normalización, que en España por ejemplo es AENOR (Agencia Española de Normalización), en Argentina el IRAM (Instituto Argentino de Racionalización de Materiales) y en Estados Unidos el ANSI (American National Standards Institute).

No conformidad – Es una condición del producto (durante cualquier etapa del desarrollo, de las pruebas o de la liberación) en la cual no se está cumpliendo con la calidad esperada, para lograr dicha calidad es necesario atender en su totalidad las no conformidades.

Normas ISO - Normas que regulan la calidad de los bienes o de los servicios que venden u ofrecen las empresas, así como los aspectos ambientales implicados en la producción de los mismos. Tanto el comercio como la industria tienden a adoptar normas de producción y comercialización uniformes para todos los países, es decir, tienden a la normalización. Ésta no sólo se traduce en leyes que regulan la producción de bienes o servicios sino que su influencia tiende a dar estabilidad a la economía, ahorrar gastos, evitar el desempleo y garantizar el funcionamiento rentable de las empresas.

Entre las normas que ha dictado esta organización se encuentran las ya conocidas ISO 9000 e ISO 14000 que son independientes una de la otra, es decir, no por tener la calificación ISO 9000 se obtiene automáticamente la ISO 14000. La ISO 9000 es el modelo de diseño-desarrollo del producto, su proceso de producción, instalación y mantenimiento, es decir, es un sistema para asegurar la calidad. Este sistema obliga a una estrecha relación entre el cliente y el proveedor; también interrelaciona cada una de las áreas de la compañía o empresa y minimiza el factor de error en la toma de decisiones en toda la organización, ya sea en situaciones habituales o especiales. Actualmente la ISO 9000 tiene más de 70.000 registros en todo el

mundo, lo cual evidencia que la comunidad de negocios internacional la ha adoptado como un sistema válido, fiable y realizable.

En 1993 la ISO comenzó en Ginebra el proceso de desarrollo de estándares de manejo ambiental para las empresas dedicadas al comercio internacional, es decir, sistemas de protección al medio ambiente que se pudieran aplicar en las empresas independientemente de condicionantes locales, regionales o estatales, e incluso del tamaño de la organización. Esto significa que el esfuerzo realizado es comparable en cualquier lugar del mundo. Por ello nace la ISO 14000, que es un sistema de estándares ambientales administrativos. Los estándares pueden ser aplicados o implementados en toda la organización o sólo en partes específicas de la misma (producción, ventas, administración, transporte, desarrollo, etc.). No hay una actividad industrial o de servicios específica a la que se pueda aplicar esas normas.

Su adopción obliga a la empresa a intentar disminuir los costos ambientales a través de estrategias como la prevención de la contaminación del agua y de la atmósfera. Lo primero que se debe conocer para optar a la calificación de ISO 14000 es en qué fallos incurre la empresa para saber dónde se puede mejorar. Es decir, se hace casi imprescindible que la empresa se someta a una auditoría ambiental que caracterice adecuadamente los efluentes, por ejemplo. El costo de una auditoría varía dependiendo de la actividad, siendo mayor cuanto más peligrosa o compleja es la actividad desarrollada (una empresa de curtidos que utiliza numerosos productos altamente tóxicos, frente a una panificadora). Con los resultados de ésta se puede comenzar a tomar las medidas correctoras para encuadrar al establecimiento dentro de la legislación sectorial vigente y así poder optar a la calificación.[2]

Sistema de calidad – Es el conjunto de elementos que constituyen el plan para lograr la calidad de un proceso, producto o método los cuales de una forma u otra interactúan entre ellos.

Software – Se llama así a los programas de computadoras, a las instrucciones responsables de que el hardware realice una tarea.

[2] Microsoft Encarta 2002

Como concepto general, el software puede dividirse en varias categorías basadas en el tipo de trabajo realizado. Las dos categorías primarias de software son los sistemas operativos (software del sistema), que controlan los trabajos del ordenador o computadora, y el software de aplicación, que dirige las distintas tareas para las que se utilizan las computadoras. Por lo tanto, el software del sistema procesa tareas tan esenciales, aunque a menudo invisibles, como el mantenimiento de los archivos del disco y la administración de la pantalla, mientras que el software de aplicación lleva a cabo tareas de tratamiento de textos, gestión de bases de datos y similares. Constituyen dos categorías separadas el software de red, que permite comunicarse a grupos de usuarios, y el software de lenguaje utilizado para escribir programas.[3]

SQA – Software Quality Assure. Aseguramiento de la calidad del software, este término se refiere a la metodología utilizada para lograr que un proyecto de desarrollo, actualización o mantenimiento de software alcance un estándar de calidad o que un producto o ciclo de desarrollo cumpla con lo necesario para lograr una certificación ante un organismo internacional competente.

V & V Verificación y validación – Es un proceso mediante el cual se realiza una evaluación técnica y sistemática del software y de sus productos asociados, esto con el fin de constatar que cumple con los requisitos funcionales establecidos.

ZD – Cero defectos es una norma de desempeño propuesta por Philip B. Crosby, que define que la mejor forma de lograr la calidad en una línea de producción es obtener productos sin defectos desde a la primera, evitando el retrabajo.

[3] Microsoft Encarta 2002

Capítulo 1. Establecimiento de las actividades para el aseguramiento de un proyecto de software

1.1 Conceptos y definiciones

Cada desarrollo, actualización, o proyecto de mantenimiento de cualquier área debe incluir algunas actividades de aseguramiento. El tipo, la cantidad, y la formalidad de esas actividades son decisiones del encargado de su realización, basadas en la evaluación del mismo; de sus riesgos y de su ambiente de desarrollo y operación.

Incluso, una persona desempeñando su trabajo, realiza actividades de aseguramiento implícitamente, aún un programador sin darse cuenta lo hace. Cada programador tiene su propia idea de cómo debe escribirse el código, y esta idea funciona como un estándar de codificación, para él mismo. Asimismo, cada uno de nosotros tenemos una idea de cómo debe ser escrita la documentación que necesitamos, y esto lo podemos considerar como un estándar personal de redacción. Cada programador revisa sus productos para cerciorarse de que cumplan con sus estándares internos, y esto lo podemos considerar como una revisión para el aseguramiento. Además cada quien prueba y examina su propio trabajo, y esto se puede considerar como un proceso de verificación y de validación. La lista podría extenderse, pero la idea debe quedar clara. Un proyecto para el aseguramiento de software involucra los procesos que cada programador puede aportar; pero requiere de la planeación y el establecimiento formal del mismo, más que los estándares y los procesos propios.

1.2 Adecuación del aseguramiento del software al proyecto

La planeación de un proyecto debe de ser un reflejo de las características específicas del mismo; así como de los riesgos que generan las necesidades de aseguramiento. Las características que deben ser consideradas son: la seguridad y misión crítica del software, la planeación de tiempos y presupuestos, tamaño y complejidad del

producto que se producirá, y tamaño y complejidad organizacional del personal de desarrollo.

La relación entre criticismo y aseguramiento es como uno la esperaría: cuan más crítico es el software, el esfuerzo de aseguramiento debe ser más importante y formal.

La relación de tiempos y presupuesto no es intuitiva; sin embargo, cuanto más justos son el presupuesto y el tiempo planeado, se vuelve más crítico el tener un esfuerzo de aseguramiento bien planeado y efectivo. Esto no significa que los proyectos con más recursos pueden permitirse ser poco estrictos, solamente significa que el hecho de tener recursos limitados aumenta los riesgos, los cuales se deben compensar con un fuerte programa de aseguramiento.

El tamaño proyectado de un software que se vaya a producir debe influir mucho en el nivel requerido de aseguramiento. Un proyecto grande requiere de estándares explícitos y detallados para todos los productos, esto se hace con el fin de obtener por lo menos un estándar mínimo de calidad; proveniente de ideas y experiencias de los programadores y del equipo de trabajo participante.

Además, un proyecto grande requiere de esfuerzos significativos en las actividades de prueba y verificación, las cuales deben ser planeadas y los planes deben ser seguidos al pie de la letra. En resumen, de acuerdo al tamaño de la actividad, se debe establecer un programa significativo y formal de aseguramiento, o deben aceptarse los riesgos que los productos de mala calidad traerían.

Por otra parte, un proyecto pequeño puede requerir un aseguramiento más informal, el cual se puede dejar a criterio del programador que esté desarrollando el software.

Otro factor que influye en la planeación del aseguramiento de la calidad es la estructura organizacional del proyecto.

Un grupo de trabajo pequeño y centralizado puede participar fácilmente en revisiones e inspecciones y mantenerse informado del estado de las no conformidades; ayudarse mutuamente, encontrando estándares de codificación y de documentación.

Un grupo de trabajo grande o disperso tendrá muchas ideas distintas de cómo hacer las cosas, y por consiguiente mucha mayor dificultad para expresarlas. En este caso, se necesitará un programa de aseguramiento más formal y un esfuerzo más grande de todos los integrantes del equipo de trabajo.

Una última pero no menos importante característica es la diferencia entre los requerimientos que puede tener una organización abastecedora de software y los que puede tener una organización adquisidora de software. Un abastecedor de software desarrolla directamente los productos, basándose en diseños y escribiendo código, etc., y por lo tanto necesita un programa completo de aseguramiento. Un adquirente no desarrolla software y por lo tanto puede limitar sus actividades de aseguramiento a cerciorarse de que el abastecedor de software esté cumpliendo con los métodos, estándares y calidad de los productos convenidos.

1.3 Creación de un plan para el aseguramiento del software

Un programa de aseguramiento efectivo requiere de una planeación y de un seguimiento apropiado; simplemente no puede desarrollarse junto con el proyecto. La planeación adecuada del aseguramiento garantiza que las actividades del mismo estén basadas en los requerimientos y los riesgos de calidad asociados al proyecto específico.

El propósito de crear un plan de aseguramiento del software; es documentar y/o especificar la conducta de las actividades que lo abarcarán durante un proyecto específico. Armado con la información sobre el proyecto y los recursos disponibles para el aseguramiento del software, además se necesita lo siguiente:

Que se planee el aseguramiento del software conjuntamente con la administración y planeación de la ingeniería, durante la fase de conceptualización e iniciación del proyecto.

Que se definan propiamente las actividades del aseguramiento. Por ejemplo, los estándares de diseño deben ser definidos antes de que éste inicie.

Que se complete el desarrollo de las herramientas antes de que se necesiten. Especialmente con las que se realizarán las pruebas del producto.

1.4 Consideraciones sobre la estructura del proyecto

En la planeación y en el establecimiento de un programa de aseguramiento del software, una consideración importante es la organización del mismo y la localización en dicha organización de las actividades de aseguramiento.

La experiencia en hardware y software indica que algunas funciones de aseguramiento se realizan mejor por entidades de la organización, separadas de aquellas que están realizando las actividades de ingeniería. El aseguramiento de la calidad del software (**SQA - Software Quality Assurance**) es una actividad que dentro de la organización se debe separar de las áreas que producen el software.

Administrativamente, el área encargada del aseguramiento de la calidad del software debe reportarle al encargado del proyecto. De hecho, muchas organizaciones exitosas que producen software tienen áreas de aseguramiento de la calidad, que administrativamente reportan a la alta dirección de la empresa, interactuando con el encargado del proyecto. La razón de esta separación de funciones es que el área de aseguramiento de la calidad debe ser el brazo derecho de la alta dirección y debe garantizar que los estándares y los procedimientos se estén cumpliendo en forma satisfactoria. Si él área de aseguramiento de la calidad no fuera independiente de las actividades de desarrollo del software, es claro que sería muy difícil que las evaluaciones fueran imparciales.

Además, muchas organizaciones han tenido éxito usando un equipo de trabajo independiente para las pruebas, o por lo menos durante las pruebas de desarrollo; este equipo es responsable de desarrollar los

planes, procedimientos, y casos de prueba para la aceptación de las pruebas formales. Se requiere la independencia porque dichas pruebas deben ser guiadas conforme a los requerimientos y no influenciadas por los detalles de la estructura del diseño y codificación.

1.5 Criterios de finalización

Debido a la naturaleza del software, es difícil comprobar el estado de una actividad de desarrollo o de mantenimiento, es importante, por lo tanto, definir criterios para la finalización de etapas específicas del mismo. Por ejemplo, durante la fase de implementación, se deben hacer diseños altamente detallados de los elementos del programa; si los elementos son pocos, codificarlos, y hacer pruebas unitarias de cada uno de ellos. Cuando interviene un número significativo de elementos en un programa, es difícil para cualquiera comprobar el estado de las unidades sin criterios específicos de finalización. Por ejemplo, pudiera haber un criterio que determine que cierto diseño está completo solamente después de que se ha realizado una inspección del mismo o pudiera haber un criterio que determine que un diseño está finalizado hasta que cumpla con una serie de pruebas.

El ajuste de los criterios de finalización es una actividad de la administración del proyecto, pero la auditoría de estados es una actividad del aseguramiento de la calidad del software. Con estos criterios se puede tener un claro conocimiento de los estados de las actividades planteadas. Esto es importante para los abastecedores y los adquirentes de software, y esta "auditoria de estados" es una actividad importante para garantizar la calidad del software.

1.6 Implementación del plan para el aseguramiento del software

Una vez que se han determinado las necesidades del proyecto y se ha completado la planeación del aseguramiento del software, se debe implementar el plan. Para esto se debe reclutar un grupo de trabajo entrenado y calificado en los puntos en los que sea necesario. Si no se pueden reutilizar los estándares y procedimientos de algún otro proyecto, se deben definir unos nuevos que sean adecuados a las necesidades en curso. El personal debe estar entrenado en dichos estándares y procedimientos, aunque su simple redacción no garantiza su cumplimiento. Todo lo que se ha mencionado en este párrafo es tarea de la administración del proyecto, pero el grupo de trabajo para el aseguramiento de la calidad es un recurso que debe ayudarla con sus actividades de administración.

El grupo de trabajo dedicado puramente a las actividades de aseguramiento es generalmente pequeño comparado con el personal de proyecto. Por otra parte, es importante tener gente con responsabilidades específicas en el aseguramiento, incluso si realiza algunas otras funciones dentro de la organización. Muy frecuentemente la frase "la calidad es el negocio de todos" se convierte en "la calidad es el negocio de nadie" si no se asignan las responsabilidades específicas

1.7 Sumario

El aseguramiento del software es una parte esencial del desarrollo y mantenimiento del mismo. Este aseguramiento forma parte de una triada de actividades, junto con la administración y la ingeniería de software, que en conjunto, pueden proporcionar una actividad exitosa de desarrollo, actualización, o mantenimiento de programas.

Capítulo 2. Aseguramiento de la calidad del software

2.1 Conceptos y definiciones

El aseguramiento de la calidad del software (**SQA: Software Quality Assurance**) se puede definir como el acercamiento planeado y sistemático a la evaluación de la calidad y la adherencia a los estándares, a los procesos, y a los procedimientos de los productos de software. La SQA tiene la función de asegurar que los estándares y los procedimientos se establezcan y se cumplan durante el ciclo de vida y la adquisición del software. La conformidad con los estándares y los procedimientos se evalúa con: la supervisión de los procesos, la evaluación del producto y las auditorías de desarrollo de software. Los procesos de control deben incluir puntos de aprobación del aseguramiento de la calidad, con los que se puede hacer una evaluación de la SQA del producto basándose en los estándares establecidos.

2.2 Estándares y procedimientos

Es crítico para el desarrollo del software establecer estándares y procedimientos, puesto que éstos proporcionan el marco de referencia para el desarrollo del mismo.

Los estándares son los criterios establecidos con los cuales se comparan los productos de software.

Los procedimientos son los criterios establecidos con los cuales se comparan el desarrollo y los procesos de control.

Los estándares y los procedimientos establecen los métodos prescritos para desarrollar el software; el papel de la SQA es asegurar que existen y son adecuados. También es necesario que se haga una documentación apropiada de los mismos, puesto que las actividades de: supervisión de proceso, evaluación del producto y auditoría, deben

confiar en que existen definiciones inequívocas, esto es con el fin de medir la conformidad del proyecto.

Los tipos de estándares que se pueden definir son:

1) Los estándares de documentación, especifican la forma y el contenido para la planeación, el control y la documentación del producto y proveen consistencia a través de un proyecto.

2) Los estándares de diseño especifican la forma y el contenido del diseño del producto. Proporcionan las reglas y los métodos para traducir los requisitos del software en diseño y para representarlo en la documentación del mismo.

3) Los estándares de código especifican el lenguaje de programación en el cual debe ser escrito y definen cualquier restricción en el uso de las características de dicho lenguaje. Definen además las estructuras del lenguaje legal, las convenciones del estilo, las reglas para las estructuras de datos y las interfaces, y la documentación interna del código.

Los procedimientos son pasos explícitos que se deberán realizar en un proceso; todos los procesos deben tener bien documentados sus procedimientos. Como ejemplos de procesos que necesitan tener procedimientos definidos tenemos: administración de la configuración, reporte de no conformidades y acción correctiva, pruebas, e inspección formal.

2.3 Actividades para lograr el aseguramiento de la calidad del software

Las actividades del aseguramiento de la calidad del software son: la evaluación del producto y la supervisión de los procesos. Estos se encargan de asegurar que se realice correctamente el desarrollo del software, los procesos de control descritos en el plan de la administración del proyecto y que se cumplan los procedimientos y los estándares del mismo.

Los productos se supervisan para revisar su conformidad con los estándares y los procesos se supervisan, para evaluar su conformidad con los procedimientos. Las auditorías son técnicas fundamentales que son utilizadas para realizar evaluaciones del producto y supervisiones de los procesos. Se debe hacer una revisión del plan de la administración para asegurar que los puntos para lograr la aprobación de la SQA estén incluidos en estos procesos.

La evaluación del producto es una actividad de la SQA, que asegura que los estándares se están cumpliendo. Idealmente, los primeros productos supervisados por la SQA deben ser los estándares y los procedimientos del proyecto. En la SQA se deben asegurar que los estándares definidos sean claros y alcanzables y después debe evaluar la conformidad del producto de software con dichos estándares.

La evaluación del producto debe asegurar que el software refleje los requisitos del estándar aplicable según se haya identificado en el plan de la administración.

La supervisión de los procesos es una actividad de la SQA que se encarga de asegurar que se estén siguiendo los pasos apropiados para realizar dichos procesos. La forma en que la SQA supervisa los procesos es comparando los pasos que se han realizados realmente con los que están documentados en los procedimientos.

Los métodos que serán usados por la SQA para la supervisión de procesos deben estar especificados en la sección de aseguramiento del plan de la administración.

Una técnica fundamental de la SQA es la auditoría, que vigila los procesos y/o los productos en profundidad, comparándolos con los procedimientos y los estándares establecidos. Las auditorías se utilizan para repasar procesos de la administración, procesos técnicos, y procesos del aseguramiento con el fin de proporcionar un indicador de la calidad y un estado del producto de software.

Los propósitos de una auditoría hecha por la SQA son: asegurar que: se están siguiendo los procedimientos de control adecuados, que se

está realizando y actualizando la documentación requerida, y que el informe de desarrollo refleje exactamente el estado de las actividades.

El producto resultante de dicha auditoría es un informe dirigido a la administración que consta de resultados y recomendaciones que tiene la finalidad de apegar el desarrollo a los estándares y/o los procedimientos.

2.4 Relaciones entre el aseguramiento de la calidad del software y otras actividades de aseguramiento

Algunas de las relaciones más importantes de la SQA con otras actividades de la administración y del aseguramiento se describen a continuación:

2.4.1 Supervisión de la administración

La SQA asegura que las actividades de la administración de la configuración del software (**CS**) se realicen de acuerdo con los planes, los estándares, y los procedimientos. La SQA revisa los planes de la CS para su conformidad con las políticas y los requisitos de la configuración requerida por los programas y proporciona el reporte de las no conformidades. La SQA revisa las funciones de la CS para su apego a los estándares y a los procedimientos y prepara informes de los resultados obtenidos.

Las actividades de la CS que supervisa la SQA son: el control de las líneas de base, la identificación de la configuración, el control de la configuración, la administración del estado de la configuración, y la autenticidad de la misma. La SQA también supervisa y revisa las bibliotecas del software.

La SQA asegura también que:

Se establezcan las líneas de base apropiadas y se mantengan constantemente a la mano para su uso durante el desarrollo y control subsecuente de dichas líneas.

La identificación de la configuración del software que sea constante y exacta con respecto a la enumeración o a la nomenclatura de los programas de computadora, de los módulos del software, de las unidades del software, y de los documentos asociados al mismo.

El control de la configuración se mantenga tal que la configuración del software que se utiliza en fases críticas de pruebas, de aceptación, y de entrega sea compatible con la documentación asociada.

La administración del estado de la configuración se realice de una manera adecuada, incluyendo el registro y la divulgación de los datos los cuales deben reflejar la identificación de la configuración del software, los cambios propuestos a la identificación de la configuración, y el estado de la puesta en marcha de cambios aprobados.

La autenticidad de la configuración del software esté establecida por una serie de revisiones de la configuración y por las auditorías que exhiban el funcionamiento requerido por la especificación de las necesidades y que además la configuración del software se refleje de una manera, fiel en los documentos de diseño del mismo.

Las bibliotecas de desarrollo del software provean el manejo apropiado del código, de la documentación, de los medios, y de los datos relacionados en sus varias formas y versiones a partir del momento de su aprobación o aceptación inicial y hasta los que se hayan incorporado al final.

Los cambios aprobados al software se hagan de forma correcta y constante en todos los productos, y que no se realice ningún cambio que no esté autorizado.

2.4.2 Verificación y validación

La SQA asegura las actividades de verificación y de validación (V&V) realizando revisiones e inspecciones técnicas. Más adelante se describe el papel que juega la SQA durante las pruebas formales.

El papel de la SQA en las revisiones e inspecciones es observar y participar según se necesite, y verificar que se conduzcan y documenten correctamente. La SQA también asegura que cualquier acción requerida esté asignada, documentada, calendarizada, y sea actualizada continuamente.

Las revisiones formales del software se deben realizar al final de cada fase del ciclo de vida del mismo, para identificar problemas y para determinar si el producto resuelve todos los requisitos definidos. Algunos ejemplos de revisiones formales son: la revisión del diseño preliminar (**PDR**), la revisión crítica del diseño (**CDR**), y la revisión de la preparación de la prueba (**TRR**).

La función de la revisión es vigilar en su totalidad que el producto que se está desarrollando satisface los requisitos. Las revisiones son parte del proceso de desarrollo, están diseñadas para proporcionar una decisión de si está listo o no para comenzar la fase siguiente. En revisiones formales, el trabajo real realizado se compara con estándares establecidos. El objetivo principal de la SQA en revisiones, es asegurar que se han seguido los planes de la administración y del desarrollo, y que el producto está listo para proceder con la fase siguiente de su ciclo. Aunque la decisión de proceder o no es una decisión de la gerencia, la SQA es responsable de aconsejarla y de participar en dicha decisión.

Una inspección es una examinación detallada de un producto, ya sea paso a paso o línea por línea de código para encontrar errores. Para las inspecciones la SQA asegura, por lo menos, que el proceso se ha terminado correctamente y que se ha realizado un seguimiento. El proceso de la inspección se puede utilizar para medir la conformidad con los estándares.

2.4.3 Revisión de pruebas formales

La SQA asegura que las pruebas formales del software, por ejemplo la prueba de aceptación, se realicen de acuerdo con los planes y procedimientos. Revisa la documentación de prueba por completo y su apego a los estándares; esta revisión de la documentación incluye: planes de prueba, especificaciones de prueba, métodos de prueba, e informes de prueba. La SQA supervisa el desarrollo de la misma y provee seguimiento a las no conformidades. Gracias a estás pruebas, la SQA asegura por completo el software y lo prepara para su liberación.

Los objetivos de la SQA en la supervisión de las pruebas formales del software son asegurar:

Que los métodos de prueba estén probando los requisitos del software de acuerdo con los planes.

Que los métodos de prueba sean comprobables.

Que se esté probando la versión correcta del software.

Que se sigan los métodos de prueba.

Que las no conformidades que ocurren durante la prueba (es decir, cualquier incidente no esperado por los métodos de prueba) se observen y se registren.

Que los informes de prueba sean realizados con exactitud y estén completos.

Que se realicen pruebas regresivas para asegurar la corrección de las no conformidades.

Que la resolución de todas las no conformidades ocurra antes de la liberación.

La prueba del software verifique que el producto cumpla con todos sus requisitos.

La calidad de la prueba es garantizada verificando que los requisitos del proyecto se hayan satisfecho y que el proceso de prueba esté de acuerdo con los planes y los procedimientos establecidos.

2.5 Aseguramiento de la calidad del software durante el ciclo de vida de adquisición

Además de las actividades descritas en los párrafos anteriores, existen actividades específicas de la SQA que se deben conducir durante el ciclo de vida de la adquisición del software. En la conclusión de cada fase, la concurrencia de la SQA es un elemento clave en la decisión de la gerencia para iniciar la fase siguiente del ciclo vital. Las actividades sugeridas para cada fase se describen a continuación.

2.5.1 Concepto de software y fase de iniciación

La SQA debe estar implicada en la escritura y el repaso del plan de la gerencia para asegurar que los procesos, los procedimientos, y los estándares identificados en dicho plan, sean apropiados, claros, específicos, y auditables. Durante esta fase, la SQA también realiza la tarea del aseguramiento de la calidad del plan de la gerencia.

2.5.2 Fase de requisitos del software

Durante la fase de los requisitos del software, la SQA asegura que los requisitos estén completos, sean comprobables, y estén correctamente expresados como requisitos funcionales y de interfaz.

2.5.3 Fase de diseño de la arquitectura del software

Las actividades que realiza la SQA durante la fase (preliminar) arquitectónica del diseño son:

Asegurar la adherencia a los estándares de diseño aprobados según lo señalado en el plan de la gerencia.

Asegurar que exista una matriz de pruebas de verificación y que sea actualizada oportunamente.

Asegurar que los documentos de control de interfaz estén de acuerdo con el estándar en forma y contenido.

Revisar la documentación de la revisión preliminar de diseño y asegurar que todos los elementos de acción hayan sido resueltos.

Asegurar que el diseño aprobado sea revisado por la administración de la configuración.

2.5.4 Fase de diseño a detalle del software

Las actividades de la SQA durante la fase de diseño a detalle del software son:

Asegurar que se sigan los estándares de diseño aprobados.

Asegurar que los módulos asignados se incluyan en el diseño detallado.

Asegurar que los resultados de las inspecciones del diseño se incluyan en el mismo.

Revisar la documentación de la revisión crítica del diseño y asegurar que todos los elementos de acción hayan sido resueltos.

2.5.5 Fase de implementación del software

Las actividades de la SQA durante la fase de implementación incluyen la revisión de:

Los resultados de las actividades de la codificación y del diseño incluyendo el estimado en tiempo contenido en el plan de desarrollo del software.

Estado de todos los componentes liberados.

Actividades de la administración de la configuración y las bibliotecas de desarrollo de software.

Reporte de no conformidades y su acción correctiva.

2.5.6 Fase de pruebas y de integración del software

Las actividades de la SQA durante la fase de integración y de prueba son:

Asegurar la preparación para probar todos los elementos liberados.

Asegurar que todas las pruebas estén siendo realizadas según los planes y procedimientos de prueba y que cualquier no conformidad se haya registrado y resuelto.

Asegurar que los informes de prueba estén completos y correctos.

Certificar que la prueba esté completa y que el software y su documentación estén listos para la entrega.

Participar en la revisión y la preparación de la prueba y asegurar que todos los elementos involucrados estén completos.

2.5.7 Fase de aceptación y liberación del software

Como mínimo, las actividades de la SQA durante la fase de la aceptación y de la entrega del software deben incluir el aseguramiento de una auditoría final con el fin de demostrar el funcionamiento de los elementos y confirmar que todos estos sean entregados y estén listos para su liberación.

2.5.8 Fase de ingeniería de mantenimiento y operación del software

Durante esta fase, habrá ciclos del mini-desarrollo para actualizar o para corregir el software. Durante estos ciclos de desarrollo, la SQA conduce las actividades específicas apropiadas descritas previamente en las etapas de desarrollo inicial.

2.6 Técnicas y herramientas

La SQA debe evaluar las herramientas que se vayan a emplear para lograr el aseguramiento según lo requiera el proyecto, algunas estarán disponibles comercialmente y otras deberán ser desarrolladas según se requiera. Como herramientas útiles pudiéramos incluir: listas de comprobación durante las auditorías y las inspecciones y analizadores automáticos de los estándares del código.

Otra herramienta que puede ser muy útil en esta etapa del proceso es los diagramas de causa y efecto.

Los diagramas de causa y efecto no son más que la representación gráfica del flujo del proceso o del proyecto, en este se dibuja primero una línea horizontal con terminación en flecha donde se coloca al final un problema que se debe resolver y en él se dibujan ramas las cuales representan a las posibles causas de dicho problema, en las ramas a su vez se dibujan los elementos que pueden contribuir a dicha causa.

El diagrama identifica las causas más probables de un problema, de manera que se pueda llevar a cabo una recolección posterior de datos y análisis;[4] en la figura 1 se muestra un esquema general de un diagrama de causa y efecto o diagrama de Ishikawa

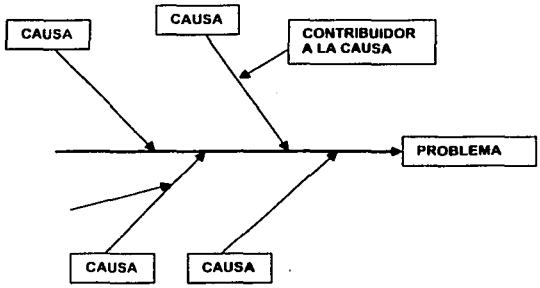

FIGURA 1

[4] Administración y control de la calidad. James R. Evans.

Capítulo 3. Ingeniería de calidad del software

3.1 Conceptos

La ingeniería de calidad del software (SQE por sus siglas en inglés Software Quality Engineering) es un proceso que evalúa, valora y mejora la calidad del software. La calidad del software se define como el grado en el que el software se apega a los requerimientos iniciales en cuanto a confiabilidad, mantenimiento, transporte, etc., en contrastaste con requisitos funcionales, de rendimiento y de interfaz que se deben satisfacer

Durante el desarrollo de un producto de software deben contemplarse la calidad para satisfacer los requerimientos establecidos por esta misma. La SQE asegura que los procesos para incorporar calidad a los productos de software se realicen apropiadamente y que el producto resultante alcance los requerimientos de calidad. El grado de conformidad con respecto a los requerimientos de calidad se debe determinar por análisis, mientras que los requerimientos de funcionalidad se demuestran a través de pruebas. La SQE realiza una función complementaria en la ingeniería de desarrollo de software. Su meta común es asegurar que se desarrolle un producto de ingeniería de software seguro, confiable y de alta calidad.

3.2 Cualidades del software

Antes de realizar una evaluación de SQE se deben seleccionar primero las cualidades que se desean evaluar. Algunas de las cualidades comúnmente usadas son: confiabilidad, mantenibilidad, transportabilidad, interoperatividad, la factibilidad de ser probado, utilidad, reusabilidad, trazabilidad, sustentabilidad y eficiencia.

3.2.1 Confiabilidad

La confiabilidad del hardware se mide en términos de promedio de tiempo ver sus fallas (MTTF por sus siglas en inglés Mean-Time-To-Failure) de un equipo dado. Para términos de software se utilizará una noción análoga, si bien los mecanismos que fallan son diferentes y las predicciones matemáticas utilizadas para el hardware no se pueden aplicar directamente al software. La confiabilidad del software se define como la relación entre el funcionamiento correcto de las funciones de software que se estén utilizando sobre un período de tiempo dado. A la ingeniería de confiabilidad del software le concierne la detección y corrección de errores del mismo; y aún más, le conciernen técnicas para compensar por errores de software desconocidos o inesperados, problemas en el hardware y en los datos del ambiente en el cual el software debe operar.

3.2.2 Mantenibilidad

La mantenibilidad del software se define como la facilidad que tiene este para encontrar y corregir errores de programación. Es algo análogo al término promedio de tiempo para reparación (MTTR por sus siglas en inglés Mean-Time-To-Repair). Aunque no haya una forma directa de medir o predecir la mantenibilidad del software, hay una cantidad significativa de atributos del software que lo hacen fácil de mantener. Entre estos se incluye la modularidad, auto documentación (documentación interna), legibilidad del código y técnicas de codificación estructurada. Estos mismos atributos también mejoran las sustentabilidad, es decir, la facilidad de mejorar el software.

3.2.3 Transportabilidad

La transportabilidad se define como la facilidad que tiene un software de poder ser llevado a un nuevo hardware y/o sistema operativo, distinto de donde fue desarrollado.

3.2.4 Interoperatividad

La interoperatividad se define como la facilidad que tienen dos o más sistemas de software para intercambiar información y mutuamente utilizar dicha información

3.2.5 Eficiencia

La eficiencia es un concepto en el cual el software utiliza un mínimo de recursos para realizar sus funciones.

Existen muchas otras cualidades del software, algunas de ellas no son importantes para un sistema específico, por lo que no requerirán evaluarlas ni mejorarlas. Maximizar alguna de las cualidades puede minimizar alguna otra, por ejemplo, incrementar la eficiencia de una pieza de software puede requerir escribir partes de código en lenguaje de ensamblador, lo que decrementará la transportabilidad y mantenibilidad del software.

3.3 Métricas

Las métricas son valores cuantitativos, usualmente se calculan en base al diseño o al código o a algún atributo del software relacionado con la calidad. Muchas métricas han sido inventadas, y la mayoría de ellas se utiliza con éxito en ambientes de desarrollo específico; pero ninguna tiene la aceptación general en el ámbito del desarrollo de software. La utilidad de las métricas es estimar el esfuerzo que conlleva a un desarrollo en términos de tiempo.

3.4 Programas de ingeniería de software

Las dos cualidades del software que requieren mayor atención son la confiabilidad y la mantenibilidad. Se han desarrollado algunos programas prácticos y técnicas para mejorarlas, aunque no puedan ser medibles ni predecibles. El tipo de actividades que se pueden incluir en un programa de SQE se describe aquí en términos de estás dos cualidades. Estas actividades pueden ser usadas como un modelo de las actividades de la SQE para alguna otra de las cualidades del software.

3.4.1 Cualidades y atributos

Un paso inicial en la planeación de un programa de SQE es seleccionar las cualidades que son importantes en el contexto de uso del software que se está desarrollando. Por ejemplo, las cualidades de más alta prioridad en un software para control de vuelos son usualmente la confiabilidad y la eficiencia. La mantenibilidad y la transportabilidad no tendrán lugar en el diseño. Por otro lado, el uso de software de análisis científico puede requerir de una alta mantenibilidad, debido a la facilidad de cambios que se requerirán, y no preocuparse mucho por la eficiencia.

Después de que se seleccionaron las cualidades del software se deben identificar los atributos específicos que pueden ayudar a incrementar esas cualidades. Por ejemplo, la modularidad es un atributo que tiende a incrementar tanto la confiabilidad como la mantenibilidad. El software modular está diseñado para ocupar porciones de código pequeñas, autocontenidas y con función única o unidades. El código modular es fácil de mantener, debido a que las interacciones entre las unidades de código son fácilmente comprensibles y las funciones de bajo nivel están contenidas en pocas unidades de código. El código modular es también más confiable, puesto que es más fácil probar una unidad pequeña y además autocontenida.

No todas las cualidades del software se pueden asociar tan fácilmente con sus atributos, la idea es seleccionar un diseño y unos atributos de software que incrementen las cualidades deseadas. Atributos como el ocultar información, la fuerza, cohesión y acoplamiento deben ser considerados.

3.4.2 Evaluaciones de calidad

Una vez que se hayan tomado algunas decisiones acerca de los objetivos de la calidad y de los atributos del software, se pueden hacer las evaluaciones de calidad. El propósito de una evaluación es medir la efectividad de un estándar o procedimiento, en función del atributo deseado del producto de software. Por ejemplo, los estándares de diseño y codificación deben ser sometidos a una evaluación. Si se desea manejar modularidad, los estándares deben definirse claramente y deben definirse estándares para el tamaño de las muestras, unidades o componentes. Debido a que la documentación interna se relaciona con la mantenibilidad, se deben definir estándares para dicha documentación, los cuales deben ser claros.

La calidad de los diseños y del código también debe ser evaluada. Esto puede realizarse como parte de un proceso de inspección o alguna auditoría de calidad. En cualquiera de los dos casos la implementación se evalúa contra los estándares y contra los conocimientos del evaluador en cuanto a prácticas de ingeniería de software.

3.4.3 Análisis de no conformidades

Una actividad de la SQE muy útil es el análisis de los registros de no conformidades. Las no conformidades deben ser analizadas para un gran número de eventos en secciones específicas o módulos de código. Si se encuentran áreas de código conteniendo un número de errores inusuales entonces este código debe ser totalmente revisado. Un alto número de errores puede deberse a una baja calidad del código, un diseño inapropiado, o requerimientos que no fueron bien

definidos o no fueron bien comprendidos. En cualquiera de los casos, el análisis puede indicar cambios o retrabajo que puede mejorar la confiabilidad del software completo. Conjuntamente con los problemas de código, el análisis puede revelar que existen desarrollos paralelos o procesos de mantenimiento que están causando esas no conformidades. Si es así, puede requerirse de una evaluación de los procedimientos la cual puede revelar que no se están siguiendo conforme al plan.

3.4.4 Ingeniería de tolerancia al fallo

Para software que debe tener una alta confiabilidad se debe establecer una actividad de tolerancia al fallo, se debe identificar el software que realiza funciones críticas, para este software la ingeniería debe determinar y desarrollar técnicas que aseguren que se alcanzará esa confiabilidad. Algunas de las técnicas que han sido desarrolladas para ambientes de alta confiabilidad son:

Revisión de los datos de entrada y tolerancia de errores. Por ejemplo, si los datos de entrada están fuera de rango o faltantes pueden afectar la confiabilidad, deben implementarse esquemas sofisticados de revisión de errores, interpolación y extrapolación para mejorar dicha confiabilidad.

Pruebas de exactitud, pueden aplicarse a porciones limitadas de código, son métodos que tiene la capacidad para demostrar que no existen errores.

Comparación de **n** elementos, este es un esquema de diseño e implementación donde un número de porciones independientes de software y de hardware operan en la misma entrada. Algún esquema de comparación se utiliza para determinar que salida usar Esto es especialmente efectivo donde se presentan errores de tiempo o de hardware.

Desarrollo independiente; en este esquema uno o más de los **n** elementos se desarrollan independientemente, esto ayuda a prevenir

la falla simultánea de todo los elementos debido a un error de codificación común.

3.5 Técnicas y herramientas

Algunas de las técnicas de tolerancia al fallo se describieron en secciones anteriores. Se pueden utilizar técnicas estadísticas para manipular las no conformidades. Además existen una gran cantidad de experimentos con técnicas de modos de falla y análisis de defectos para hardware que pueden adecuarse al software.

Existen también herramientas útiles para la ingeniería de la calidad, las cuales incluyen a sistemas simuladores, los cuales permiten modelar el comportamiento del sistema; analizadores dinámicos, los cuales detectan que porciones del código se están utilizando más intensamente; herramientas de software que son utilizadas para computar métricas a partir del código o del diseño y un grupo de herramientas de propósito especial que pueden por ejemplo detectar todas las llamadas del sistema para ayudar a determinar los límites de la portabilidad.

Capítulo 4. Verificación y validación

4.1. Conceptos y definiciones

La verificación y validación (V & V) del software es el proceso de asegurar que el producto que se está desarrollando o que va a sufrir alguna modificación va a satisfacer los requerimientos funcionales y que cada paso en el proceso de construcción del mismo cumple con su función. Las diferencias entre verificación y validación no tienen importancia excepto para los teóricos; en la práctica se usan los términos verificación y validación para referirse a todas las actividades que tienen como intención asegurar que el software va a funcionar como se requiere.

La verificación y la validación pretenden ser una evaluación sistemática y técnica del software y de sus productos asociados durante los procesos de desarrollo y mantenimiento. Las revisiones y las pruebas se hacen al final de cada fase del proceso de desarrollo para asegurar que los requerimientos se han completado y está listo para probarse y que el diseño, el código y la documentación satisfacen dichos requerimientos.

4.2 Actividades

Las dos actividades más importantes de la verificación y la validación son: las revisiones, incluyendo inspecciones y las pruebas.

4.2.1 Revisiones e inspecciones

Las revisiones se realizan durante y al final de cada fase del ciclo de vida, para determinar si se está cumpliendo con los requisitos establecidos, conceptos de diseño y especificaciones. Las revisiones son presentaciones del material que se realizan ante un grupo y son mucho más efectivas cuando son conducidas por personal que no ha

estado directamente involucrado en el desarrollo del software que se va a revisar.

Las revisiones informales se van realizando conforme se va necesitando. El desarrollador escoge el material que se va a presentar y/o lo expone ante el grupo de revisión. El material puede ser tan informal como un listado de computadora o un documento escrito a mano.

Las revisiones formales se conducen al final del ciclo de vida de cada fase. El adquiridor del software designa al grupo de revisión formal, el grupo tiene las facultades para tomar una decisión de proceder al siguiente paso del ciclo de desarrollo. Las revisiones formales incluyen: revisiones a los requerimientos del software, revisiones al diseño preliminar del mismo, revisiones al diseño crítico de éste y las revisiones de las pruebas de finalización de cada fase.

Una inspección es una examinación detallada de un producto de software, ya sea paso a paso o línea por línea de código. El propósito de realizar inspecciones es encontrar errores. El grupo que conduce una inspección realiza actividades de: desarrollo, pruebas y aseguramiento de la calidad del software.

4.2.2 Pruebas

Una prueba se define como la operación del software ya sea con entradas reales o simuladas, esto con el fin de demostrar que el producto satisface sus requerimientos y si no lo hace, identificar las diferencias específicas entre los resultados actuales y los resultados que se esperaban. Existe una gran variedad de pruebas de software de varios niveles, que van desde pruebas unitarias o pruebas por elemento hasta pruebas de integración o pruebas de rendimiento pasando por pruebas de software o pruebas de aceptación.

4.2.2.1. Pruebas informales

Las pruebas informales son realizadas por el desarrollador para medir el progreso del desarrollo. "Informal", en este caso no significa que las pruebas se hagan de una manera casual, solo que el adquiridor del software no está formalmente involucrado, que la testificación de las pruebas no es requerida y que el propósito principal es encontrar errores. Se consideran informales las pruebas unitarias de componentes y de integración de subsistemas.

Las pruebas informales pueden ser conducidas por requerimiento o por diseño. Las pruebas por requerimiento o pruebas de "caja negra" se hacen seleccionando los datos de entrada y otros parámetros basados en los requerimientos del software y observando las salidas y reacciones del mismo. Las pruebas de "caja negra" pueden ser realizadas en cualquier nivel de integración. Además de probar la satisfacción conforme a los requerimientos otros objetivos de este tipo de pruebas son los siguientes:

Correcciones computacionales.

El manejo correcto de las condiciones límite, incluyendo entrada de datos extremas y condiciones que causen salidas extremas.

Estados de transición según lo estimado.

El comportamiento apropiado bajo presión y bajo condiciones de mucha carga.

Detección adecuada de errores, manejo y recuperación.

La prueba conducida por diseño o prueba de "caja blanca" es el proceso en el cual la persona que hace la prueba examina los procesos internos del código. Este tipo de pruebas se hacen seleccionando los datos de entrada y otros parámetros basados en la lógica interna que va a ser revisada. Las metas de las pruebas conducidas por diseño incluyen:

Todas las rutas a través del código. Para la mayoría del software esto puede únicamente hacerse a través de una prueba a nivel unitario.

Pruebas del funcionamiento de las interfaces bit a bit.

Dimensionamiento en tiempo y tamaño de elementos de código críticos.

4.2.2.2 Pruebas formales

Las pruebas formales tienen como propósito demostrar que el software está listo para realizar las funciones para las cuales fue hecho. Una prueba formal debe incluir un plan aprobado por el adquiridor del software, procedimientos bien establecidos, testificación del aseguramiento de la calidad, un registro de todas las discrepancias y un reporte de las pruebas. Las pruebas formales son siempre conducidas por requerimiento y su propósito es demostrar que el software cumple con todos ellos.

Cada proyecto de desarrollo de software debe tener al menos una prueba formal, la prueba de aceptación que concluye las actividades de desarrollo y demuestra que el software está listo para operar.

Además de la prueba final de aceptación se puede realizar alguna otra prueba formal al proyecto. Por ejemplo, si el software se está desarrollando y liberando en forma proporcional hasta finalizar; puede haber pruebas de aceptación incrementales. De forma práctica cualquier prueba contractualmente requerida se considera como formal, cualquier otra se considera como "informal".

Después de la aceptación de un producto de software, todos los cambios al mismo deben ser aceptados solo como resultado de una prueba formal. Las pruebas en la etapa de post aceptación deben incluir pruebas de regresión. Una prueba de regresión consiste en realizar prueba que ya se habían hecho, para asegurar que el cambio no afecta otras funcionalidades que ya habían sido aceptadas.

4.3 Verificación y validación durante el ciclo de vida de adquisición del software.

La verificación y la validación del ciclo de vida del software las podemos dividir en las siguientes fases:

4.3.1 Concepto de software y fase de inicialización

La mayor actividad de la verificación y la validación (V & V) es desarrollar un concepto de cómo se revisará y probará el sistema. En los proyectos simples los pasos del ciclo de vida pueden compactarse; si eso pasa, las revisiones pueden compactarse también. Los conceptos de prueba pueden involucrar la simple generación de casos de prueba por un usuario representativo o pueden requerir el desarrollo de simuladores elaborados o generadores de datos de prueba. Sin un concepto adecuado de verificación y validación además de un plan, no se puede tener un estimado apropiado de costos y tiempos y por consiguiente no se puede estimar la complejidad del proyecto.

4.3.2 Fase de requerimientos del software.

Durante esta fase las actividades de la verificación y validación deben incluir:

El análisis de los requerimientos del software para determinar si son consistentes con el ámbito del sistema.

Asegurar que los requerimientos son comprobables y capaces de ser satisfechos.

Crear una versión preliminar del plan de pruebas de aceptación, incluyendo una matriz de verificación, la cual relaciona los requerimientos con las pruebas usadas para demostrar que los mismos pueden ser satisfechos.

Comenzar el desarrollo de simuladores y generadores de datos de prueba si fueran necesarios.

La revisión de los requerimientos al final de cada fase.

4.3.3 Fase de diseño de la arquitectura del software.

Durante esta fase las actividades de la verificación y validación deben incluir:

Actualización de la versión preliminar del plan de pruebas de aceptación y de la matriz de verificación.

La realización de revisiones informales o inspecciones del diseño preliminar del software y de la base de datos.

La revisión preliminar del diseño al final de la fase en la cual se inspeccionan y aprueban los requerimientos de la arquitectura del software.

4.3.4 Fase de diseño del detalle del software.

En esta fase se deben realizar las siguientes actividades:

Completar el plan de pruebas de aceptación y la matriz de verificación incluyendo las especificaciones de las pruebas y los planes de pruebas unitarias.

Realizar revisiones informales o inspecciones a detalle del diseño del software.

La revisión crítica del diseño la cual completa la fase del detalle del software.

4.3.5 Fase de implementación del software.

Durante esta fase (V & V) debe realizar las siguientes funciones.

Inspecciones del código.

Pruebas unitarias del software y estructuras de datos.

Localización, corrección y pruebas de errores.

Desarrollo de procedimientos detallados de pruebas para las siguientes dos fases.

4.3.6 Fase de pruebas y de integración del software.

Esta fase es la que requiere mayor esfuerzo por parte de la verificación y la validación, donde las pruebas unitarias de la fase previa se integran en subsistemas y en seguida al sistema final. Las actividades durante esta fase deben incluir:

La realización de pruebas a procedimientos de prueba.

La documentación de: pruebas de funcionamiento, pruebas de finalización y los resultados de las pruebas de conformidad contra lo estimado.

Proveer un reporte de pruebas que incluya un sumario de no conformidades encontradas durante las mismas.

Localizar, registrar, corregir y volver a probar las no conformidades.

Confirmar que el producto está listo para las pruebas de aceptación.

4.3.7 Fase de aceptación y liberación del software.

En esta fase se realizan las siguientes actividades:

Demostrar que el software alcanza sus requerimientos funcionales, de rendimiento y de interfaz a través de pruebas, análisis e inspecciones.

Localizar, corregir y volver a probar las no conformidades.

4.3.8 Fase de ingeniería de mantenimiento y operación del software.

Durante esta fase post liberación (V & V) puede volver a realizar alguna de las actividades que se realizaron durante las siete fases anteriores cuando sea requerido por alguna actualización del software.

4.4 Verificación y validación independientes.

La verificación y validación independientes (IV & V) es un proceso donde se revisan, verifican y validan independientemente las fases del ciclo de vida de los productos de software, esta acción la realiza una organización que no forma parte ni de los desarrolladores ni de los adquiridores del software. El agente que realiza la (IV & V) no debe estar ni a favor ni en contra del éxito del software, su único propósito debe ser asegurarse que el mismo sea probado contra una serie de requisitos.

Las actividades del proceso de verificación y validación independientes duplican paso a paso las actividades de (V & V) durante el ciclo de vida, con la excepción de que la (IV & V) no realiza pruebas informales. El agente que realiza la (IV & V) sólo puede realizar en una ocasión las pruebas formales de aceptación. En este caso el desarrollador debe hacer una demostración formal de que el software está listo para su aceptación.

4.5 Técnicas y herramientas.

Tal vez se hayan desarrollado más herramientas para ayudar a las actividades de (V & V) de software (especialmente para pruebas) que para cualquier otra actividad relacionada con el mismo. Las herramientas disponibles incluyen: seguidores de código, analizadores de memoria de propósito especial, generadores de datos, simuladores y emuladores. Algunas herramientas son esenciales para las pruebas del software al grado de ser determinantes para el costo y el tiempo de desarrollo.

Una técnica especialmente útil para encontrar errores es la inspección formal. Las inspecciones formales fueron desarrolladas por Michael Fagan de IBM. Las inspecciones involucran evaluaciones línea por línea del producto que se está revisando. Las inspecciones son hechas por un equipo donde cada integrante realiza un papel específico. El equipo tiene como líder a un moderador, quien debe estar entrenado en procesos de inspección formal. El equipo incluye un lector quien guía al equipo a través del código, una o más personas que revisan el código, las cuales tienen como función encontrar fallas, una persona que lleva un registro de las mismas y el autor, que se encarga de explicar el elemento de software que se está inspeccionando.

Este proceso de inspección formal altamente estructurado ha resultado muy efectivo para encontrar errores y eliminarlos. Puede ser aplicado a cualquier proceso de desarrollo de software incluyendo documentación, diseño y codificación. Uno de sus beneficios es la retroalimentación directa entre el desarrollador y el autor y el mejoramiento significativo de la calidad de los resultados.

Capítulo 5. Reporte de no conformidades y acciones correctivas.

5.1 Conceptos y definiciones

El propósito de un sistema o procedimiento de reporte de no conformidades y acciones correctivas (NRCA por sus siglas en inglés Nonconformance and Corrective Actions) es reportar, analizar y corregir no conformidades. Una no conformidad puede ser considerada como un problema, una discrepancia, una anomalía, una falla, un error, cualquier error en la documentación del software, en el código, o en la estructura de datos, o cualquier evento que no cumpla con los requerimientos o los estándares. La acción correctiva es el nombre genérico que se le da al proceso por el cual se corrigen y controlan las no conformidades.

La necesidad de un sistema NRCA surge en el ciclo de vida del software, tan pronto como se desarrollan los primeros documentos y otros productos. Un sistema NRCA debe dar seguimiento a las no conformidades asignar prioridades, registrar sus disposiciones, anotar la versión del producto en el cual se están corrigiendo dichas no conformidades, notificar al originador de la no conformidad acerca de las acciones que se están tomando y producir reportes de administración.

5.2 Actividades

Las actividades las podemos dividir en:

5.2.1 Detección y reporte de no conformidades

Por definición una no conformidad es la desviación que tiene cualquier producto con respecto a los requerimientos o estándares definidos. Un reporte de no conformidades puede ser especificado para cualquier producto, en cualquier fase del ciclo de vida del software y por

cualquier persona asociada al proyecto. Normalmente, el sistema NRCA se utiliza cuando el desarrollador da su visto bueno y el sistema está listo para utilizarse ampliamente. Por ejemplo, Mientras que el desarrollador está probando unitariamente su código, los errores encontrados pueden ser seguidos localmente. Después de que se declara el código como corregido y está listo para su integración con los módulos restantes es cuando se aplica el sistema de no conformidades para informar a los usuarios del código acerca de las mismas y asegurar que se corrigieron y no se pasaron por alto.

Usualmente se utiliza una forma especial para realizar el reporte de no conformidades, estos son algunos ejemplos de la información que puede contener dicha forma:

Fecha y hora de la detección de la no conformidad.

Código de identificación del error (número de reporte y título).

Reporte individual y organización.

Individuo responsable de la acción correctiva.

Importancia de la no conformidad.

Extracto de la no conformidad.

Solución propuesta para la no conformidad.

Identificador de la unidad de código o documento de donde puede obtenerse la acción correctiva.

Fase del ciclo de vida del software en la cual surgió la no conformidad.

Solución final para la no conformidad.

Fecha y versión del elemento en el cual se corrigió la no conformidad.

Fecha en la cual se dio por cerrada la no conformidad.

5.2.2 Seguimiento y administración de reportes.

Después de que el individuo que encontró la no conformidad preparó el reporte, los datos son introducidos en un sistema de control de formas. Pueden utilizarse sistemas de bases de datos para ayudar a automatizar el laborioso esfuerzo de dar seguimiento a las no conformidades y así poder realizar reportes para la administración.

Un sistema de reporte y seguimiento de no conformidades debe tener la capacidad de producir reportes para la administración que contengan: el error y el estado de la corrección, el número de errores encontrados por producto y la criticidad de los problemas abiertos. Deben ser evaluados estos datos para determinar el impacto que tiene la no conformidad y así priorizar recursos.

5.2.3 Evaluación de impacto y acciones correctivas

Los reportes de no conformidades deben evaluarse por su nivel crítico y su nivel de importancia; se deben considerar los siguientes factores:

El impacto que tendría no corregir la no conformidad.

Los recursos requeridos para corregir la no conformidad.

El impacto en algún otro elemento si se corrige la no conformidad.

Si la decisión que se toma es la de corregir la no conformidad, deben existir procedimientos para controlar los procesos de la acción correctiva. Tales procedimientos deben incluir un seguimiento para asegurarse de que la no conformidad ha sido corregida y documentada en la versión apropiada del software y asegurarse que se realizaron las pruebas adecuadas, incluyendo pruebas de regresión.

5.3 Interrelaciones

La NRCA es una herramienta básica y fundamental para la administración del proyecto y para el aseguramiento de la calidad del software. La administración debe dar seguimiento a los cambios del producto que se generen, así como a sus versiones que resulten de la corrección de las no conformidades, además algunos reportes de las mismas contendrán cambios a los requerimientos, lo cual traerá como resultado un cambio de expectativas.

Si la no conformidad se encuentra en el código o en los datos del producto, es responsabilidad de las actividades de verificación y validación desarrollar pruebas para asegurarse que el problema se ha resuelto satisfactoriamente, además son necesarias pruebas de regresión para asegurar que no surgieron nuevos problemas con la corrección.

El aseguramiento de la calidad del software debe especificar que se siguieron al pie de la letra los procedimientos en el proceso de no conformidades, debe asegurarse también que el producto modificado aún se encuentra dentro de los estándares establecidos. El aseguramiento de la calidad del software puede solicitar una auditoría si así lo considera necesario a alguna fase determinada del ciclo de vida o a algún elemento específico del sistema.

El sistema NRCA es una herramienta muy útil para la ingeniería de la calidad del software debido a que un producto que presenta muchas no conformidades tiene una calidad muy pobre por lo que su confiabilidad necesita ser reexaminada. En sistemas en los cuales la seguridad es un requisito indispensable, existirá un grupo de seguridad que examine las no conformidades para asegurar que no se vean comprometidos los requerimientos y buscar problemas que puedan surgir de improvisto.

5.4 Técnicas y herramientas

Cada proyecto debe considerar el como utilizar un sistema automático de seguimiento de no conformidades, el cual emita reportes y haga actualizaciones automáticas de los estados de las mismas, además de registrar los cambios al producto provenientes de la resolución de las no conformidades.

Además para tener una idea de la cantidad y del tipo de las no conformidades se puede hacer uso de los diagramas de Pareto.

Un diagrama de Pareto es un histograma de los datos obtenidos, desde la frecuencia más elevada hasta la más baja[5]. Con esto es muy común determinar que se tiene una relación 80/20 lo que significa que el 80 por ciento de las no conformidades es provocado por el 20 por ciento de las causas.

Con esto se deduce que eliminando la minoría de las causas eliminamos la mayoría de las no conformidades.

Administración y control de la calidad James R. Evans.[5]

Capítulo 6. Seguridad del software y de sistemas.

6.1 Conceptos y definiciones

La seguridad del software va de la mano con la posibilidad de una falla catastrófica de los sistemas que comprometa la seguridad de las personas o propiamente resulte en una falla en la misión que se tenía planeada. La seguridad del software se define únicamente en el contexto de sistemas. El software no tiene peligros inherentes; sin embargo, .los sistemas controlados o supervisados por software fallan y algunas fallas de algunos sistemas tienen impacto en la seguridad. Para prevenir que el sistema presente fallas de este tipo existe una actividad llamada "seguridad del software".

Si estamos considerando que la seguridad del software únicamente se presenta en el contexto de sistemas, debemos entonces considerar también las no conformidades y los requerimientos del mismo. Ciertamente, los problemas más serios con los sistemas basados en software son los que se presentan cuando el desarrollo se realizó con requerimientos incorrectos, inapropiados o incompletos en base a la situación del sistema.

6.2 Problemas de software

Las fallas que pueden llegar a presentar los sistemas pueden deberse a dos tipos de problemas de software: no conformidades (o fallas que no satisfacen los requerimientos) o un error u omisión en los requisitos del software. Una no conformidad puede ser simple (las más comunes son errores de codificación o "bugs" como se conocen en el ámbito del desarrollo de software) o más compleja (por ejemplo un error en el sistema que retrase el lanzamiento de un cohete espacial). Un punto importante acerca de las no conformidades es que las técnicas de verificación y validación están diseñadas para detectarlas y existen también técnicas de aseguramiento para prevenirlas, mejoras a estos

métodos y un programa de seguridad basado en aplicaciones específicas están mejorando la seguridad y confiabilidad de sistemas de control basados en software.

Un error u omisión en los requisitos es menos detectable. El software puede realizar la función exacta que se le requirió pero el sistema puede llegar a presentar un estado que no estaba previsto. Cuando el sistema entra en un estado no definido se puede presentar un comportamiento inesperado o indeseable. Este tipo de problema no puede ser manejado por la disciplina del software; es el resultado de una falla del sistema y de los procesos de ingeniería del software que plasmaron los requerimientos iniciales en el mismo.

6.3 Métodos para mejorar la seguridad del software.

El mejoramiento en el proceso de desarrollo del software y la forma de construirlo mejor, son formas de mejorar la confiabilidad del sistema, produciendo software con menos fallas. Por consiguiente, un software más confiable es probablemente en software más seguro, pero desde el punto de vista de la seguridad para tener un sistema seguro deben realizarse algunas actividades extras. Una primera aproximación es identificar el software crítico que controla las funciones del sistema relacionadas con la seguridad y que pone especial atención a través de los procesos de desarrollo y pruebas. Este es solamente un caso especial de un método de cómo construir mejor pero enfoca pocos recursos en áreas críticas.

6.4 Programa de seguridad del software (ejemplo).

Un análisis aleatorio del sistema puede indicar que algún software requiere un programa de seguridad más formal porque incluye un componente del sistema crítico para la seguridad. El programa de seguridad para el software comienza con un análisis preliminar, el propósito de este análisis es identificar las funciones controladas por software que puedan afectar a la seguridad del mismo, cuando se identifica un componente crítico se inician las actividades de seguridad

en dicho componente dándole mucho más énfasis a las fases de codificación, pruebas y en general a todo su desarrollo.

6.4.1 Análisis de requerimientos.

El análisis de los requerimientos de seguridad del software forman la base para las actividades subsecuentes en cuestión de seguridad, este proceso evalúa tanto los requisitos del mismo como los requisitos de interfaz; y además, intenta identificar errores y deficiencias en los mismos que puedan dar como resultado la presentación de estados aleatorios en los cuales pueda caer el sistema. Algunas de las técnicas empleadas para realizar el análisis de requisitos incluyen: análisis de criticidad, análisis de especificaciones y análisis de tiempos y tamaños.

El análisis de criticidad evalúa cada requerimiento en términos de los objetivos de seguridad derivados de un componente de software dado. Esta evaluación sirve para determinar qué requerimientos tienen implicaciones de seguridad; si es así, este requerimiento debe ser considerado como crítico y se le debe dar un seguimiento a través del ciclo de desarrollo del software; esto significa, a través del diseño, codificación y pruebas.

El análisis de especificaciones evalúa que los requisitos críticos para la seguridad estén completos, que sean correctos, que tengan consistencia y que sean comprobables, este análisis considera cada requisito como único.

El análisis de tiempos y tamaños evalúa los requerimientos del software relacionados con el tiempo de ejecución, cantidad de memoria reservada y utilización del canal de comunicación. Este tipo de análisis establece y determina los tiempos de ejecución máximos permitidos; así como la utilización máxima y la disponibilidad de la memoria del sistema.

6.4.2 Análisis del diseño

El análisis del diseño verifica que el diseño del programa implemente correctamente los requerimientos de seguridad.

El análisis de diseño lógico evalúa las ecuaciones, algoritmos y lógica de control del diseño del software.

El análisis de los datos del diseño evalúa la descripción y el uso pretendido para cada elemento de datos utilizado en el diseño de un componente crítico. Las interrupciones y sus efectos deben tener atención especial en áreas críticas para la seguridad, se debe verificar que las mismas o las rutinas que las manejan no alteren datos utilizados por otras rutinas.

El análisis del diseño de la interfaz verifica que la misma realice propiamente sus funciones con otros componentes del sistema incluyendo hardware, software y operadores.

El análisis de restricciones del diseño evalúa las soluciones del mismo contra las restricciones impuestas por los requisitos y las limitaciones del mundo real. El diseño debe ser responsable de conocer y anticiparse a las restricciones que puede tener el componente de software. Estas restricciones pueden incluir: tiempo de ejecución, tamaño de los componentes y de la memoria, ancho de banda, limitaciones en las ecuaciones y en los algoritmos, limitaciones en los datos de entrada y salida y limitaciones en el diseño de la solución.

6.4.3 Análisis del código

El análisis del código verifica que el programa codificado implemente propiamente el diseño y no viole los requisitos de seguridad. Las técnicas que se utilizan en este análisis pueden ser similares a las utilizadas en el análisis del diseño.

6.4.4 Pruebas de seguridad

Las pruebas de seguridad verifican los resultados de los análisis, investiga el comportamiento del programa y confirma que el programa cumple con los requisitos de seguridad, en acuerdo con el plan de pruebas de seguridad y con los procedimientos establecidos. Las pruebas de seguridad se enfocan en encontrar debilidades en el programa e identificar situaciones extremas o inesperadas que puedan causar que el software falle y cause violaciones a los requisitos de seguridad. Este tipo de pruebas se limitan a aquellos requisitos del software considerados como críticos.

6.5 Técnicas y herramientas.

En los últimos años ha habido muchos esfuerzos para adaptar métodos usados en la seguridad del hardware a la confiabilidad del software. Herramientas como el análisis de árbol de fallas han sido aplicadas al software con cierto éxito. Se han realizado modelos de software utilizando redes de Petri y algunas otras técnicas; pero solo se ha obtenido un éxito parcial hasta la fecha. Algunas técnicas tienen una utilidad limitada y su éxito depende en gran parte de la habilidad del analista que la aplica.

Capítulo 7. Aseguramiento de la seguridad del software

7.1 Conceptos y definiciones

Cada empresa debe tener una política con respecto a la seguridad del software, dicha política debe contemplar un cierto nivel de seguridad y de integridad, el cual debe ser consistente con el daño potencial de la pérdida de información, su inexactitud, su alteración, su falta de disponibilidad y su falta de uso. El aseguramiento de la seguridad del software es el proceso que garantiza que los requerimientos de seguridad previamente establecidos durante todas las fases del ciclo de vida se están cumpliendo.

7.2 Actividades para asegurar la seguridad del software

Un programa de aseguramiento de la seguridad del software debe requerir como mínimo lo siguiente:

Realizar una evaluación de seguridad de los componentes críticos.

Que hayan sido establecidos los requisitos de seguridad para el proceso de desarrollo y/o mantenimiento.

Que hayan sido establecidos los requisitos para el software y los datos que se están desarrollando y/o manteniendo.

Que cada revisión y/o auditoría incluya una evaluación de los requisitos de seguridad.

Que los procesos de las acciones correctivas de las no conformidades provean seguridad para el software existente y que la evaluación de los cambios prevenga violaciones de seguridad.

Que sea adecuada la seguridad física tanto del software como de los datos.

Conclusiones

De la metodología expuesta a lo largo del presente trabajo, se puede concluir que es necesario establecer una estrategia que haga posible el aseguramiento tanto de la calidad, como de la seguridad de los sistemas informáticos; en efecto, podemos decir que: "la calidad nos lleva a la productividad, y viceversa". Esto significa que sólo será posible la competencia en tanto exista convergencia entre la calidad y la productividad. Por lo tanto, estimamos que los procedimientos aquí presentados permiten cumplir con los lineamientos necesarios para el logro de la calidad en los sistemas de información, la cual pude conducir a un proceso de certificación. Lo anterior, redunda en la satisfacción y la seguridad de la información con la que trabaja el usuario.

Dicho proceso de certificación se basa en el cumplimiento de normas, las cuales se establecen a partir de un estándar determinado por los organismos reguladores, que aceptan los lineamientos propuestos por la empresa a la que se destinen los sistemas informáticos en que se aplique la metodología propuesta, al cubrirse cada uno de sus puntos.

Es crucial la mejora continua de esta metodología, tanto a mediano como a largo plazo, para las empresas sometidas a las presiones del mercado (sobre todo considerando la influencia de la globalización). Tal vez pueda hacerse la excepción en lo que respecta a las empresas dedicadas a la manufactura, debido a que éstas rara vez dependen de los sistemas informáticos basados en software.

Finalmente, cabe mencionar que, la metodología presentada dota de excelentes herramientas, aptas para el logro del aseguramiento de la calidad en los sistemas informáticos, escalando a la certificación si así se desea, llevándose a cabo a partir de las disposiciones de un organismo regulador, sobre todo en el caso de empresas que integran protocolos de aceptación dentro de sus normas, que les permitan actuar dentro de cierto rango y, en consecuencia, ofrecer productos de mayor calidad y precios más competitivos en el mercado, que estarán regidos por los estándares de calidad más estrictos, en comparación con empresas del mismo ramo carentes de dichos parámetros de calidad en sus sistemas informáticos.

Llegando tal vez a cumplir con las teorías idealistas que proponía el Doctor Phillip Crosby de lograr **"Cero defectos" (ZD)** en sus productos.

Bibliografía

Software Assurance Guidebook and Standard NASA-GB-A201.
Software Assurance Technology Center.
National Aeronautics and Space Administration. N.A.S.A.

Software Quality Assurance Audits Guidebook.
Software Assurance Technology Center.
National Aeronautics and Space Administration. N.A.S.A.
Noviembre 1990.

Software Assurance Standard.
NASA-STD-2201-93.
Software Assurance Technology Center.
National Aeronautics and Space Administration. N.A.S.A.
Frederick Gregory.
Associate Administrator for Safety and Mission Quality.
Noviembre 1992.

Administración y Control de la Calidad.
Cuarta edición.
James R. Evans.
William Lindsay.
Thomson Editores.
2000.

Microsoft Encarta 2002
1993-2001 Microsoft Corporation
Encarta es una marca registrada de Microsoft Corporation.