UNIVERSIDAD NACIONAL AUTÓNOMA DE

MÉXICO

FACULTAD DE INGENIERIA

“USO DE LOS ESTÁNDARES DE LA IEEE PARA EL DESARROLLO DE SOFTWARE”

T E S I S

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACIÒN

P R E S E N T A :

A L I C I A G U A D A L U P E L O B A T O P A C H E C O

Director de Tesis: MC. REYNALDO ALÀNIS CANTÙ

México, D. F. **AGOSTO 2005**

Gracias Dios por darme vida y permitir dar a mi familia este regalo.

Dedico esta tesis a:

A la mejor amiga y consejera que he tenido en esta vida, a esa persona incondicional que ha estado en las buenas y en las malas conmigo, que me ha dado su apoyo y todo su amor, a esa persona que me ha enseñado a vivir, a creer, a amar, a perdonar y a luchar por lo que se quiere, y sobretodo, le ha dado un completo y total sentido a mi vida, sin ella no sería lo que soy.

¡¡¡Lo logré mamá!!!

A ti papá por el apoyo total para la realización de esta tesis, y por el ejemplo de honestidad y responsabilidad que me has enseñado siempre en el ámbito laboral.

A ti Gaby por ser la mejor hermana del mundo, por compartir muchísimos momentos juntas. Yo te admiro por lo que has logrado y recuerda que si se quiere, todo se puede.

A ti Memo por ser una personita muy inteligente y por haber sido para nosotros una luz en nuestras vidas.

A ti Nala, aunque no tengas las orejas de la familia ni la nariz, has formado ya parte de mi y eres un ser muy especial.

Los amo con todo el corazón

Hay personas muy especiales para mí, que ya no están en este mundo, pero si en mi corazón: Abuela Esther, Abuelo Nicolás, Abuelo Guillermo y Luis.
Dedico de manera muy especial esta tesis a mi prima Bibiana, pues le prometí hacer cosas “fregonas” como las que ella hizo y las que le faltaron por hacer. Gracias Bibiana por enseñarme que la vida se vive intensamente.

A todos mis primos por ser antes que primos amigos, y haber crecido conmigo. Agradezco haber conocido muchas cosas con ustedes y haber vivido experiencias inolvidables.

A ti Patita porque siempre he admirado tu inteligencia, honestidad y fortaleza. Tus consejos siempre fueron importantes para mí.

A mis tíos, por motivarme constantemente a concluir mi carrera y titularme, además por formar una gran familia (aunque no sea perfecta), unida en las buenas y en las malas.

A mis amigas Roxana y Adriana a las cuales admiro mucho y en breve espero pertenecer a su círculo de triunfadoras. Con ustedes descubrí que existe la verdadera amistad incondicional y para siempre.

A ti Sergio por darme tu apoyo para realizar esta tesis, por aguantar mis histerias, por echarme porras cuando mi miedo quería vencerme y por demostrarme que no tiene nada que ver la edad con el amor, la inteligencia y la responsabilidad. Te admiro mucho por lo que eres y en donde estás a pesar de todo, sigue adelante y por nada ni nadie caigas de nuevo.

Los quiero mucho a todos

Agradezco a mi asesor de tesis, el MC. Reynaldo Alánis Cantú, por creer en mí y apoyarme para realizar esta tesis.

“Gracias profesor por no dejar que “tirara la toalla” cuando pensé que no podría”

Muchas Gracias

ÍNDICE

INTRODUCCIÓN

Uno de los problemas que se afrontan actualmente en la esfera de la computación es la calidad del *software*. Desde la década del los 70's, este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de *software*, los cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales, ¿cómo obtener un *software* con calidad? y ¿cómo evaluar la calidad del *software*?, ambas interrogantes llevan amplias respuestas, pero están estrechamente ligadas con el concepto de la calidad del *software*, que es el resultado de la primera y la fuente de la segunda.

El software es inmaterial y la calidad del software difícil de medir, pero tenemos algunas pautas, algunos indicadores que nos ayudan a diferenciar los productos de calidad de los que carecen de ella:

- El acercamiento a cero defectos.
- El cumplimiento de los requisitos intrínsecos y expresos.
- La satisfacción del cliente (sobre todo la satisfacción del cliente).

El argumento de la calidad es exhibido por las empresas como un factor diferenciador, como clave de sus procesos de negocio y como eslogan de competitividad empresarial. De hecho, la exigencia cada vez mayor por parte del mercado de la certificación ISO 9000 o el interés creciente por los modelos de calidad de gestión empresarial son indicadores de la percepción de la calidad como un elemento cada vez más necesario.

El principal instrumento para garantizar la calidad de las aplicaciones sigue siendo el Plan de Calidad. El plan se basa en unas normas o estándares genéricos y en unos procedimientos particulares.

Los estándares se desarrollan por grupos de individuos u organizaciones para armonizar las especificaciones de productos, interfaces, procesos, terminología, etc., cubren una amplia gama de tópicos y son reconocidos por varios grupos de personas y países.
Algunos estándares se desarrollan formalmente por organizaciones mientras que otros se imponen en el mercado.

Cuando se habla de estándar, nos referimos a una regla o base de comparación que se utiliza para medir algún aspecto del software, como la calidad, productividad, duración, esfuerzo o costo. Se puede tomar como una definición más estructurada la siguiente:

> "Un estándar es un conjunto de criterios aprobados, documentados y disponibles para determinar la adecuación de una acción (estándar de proceso) o de un objeto (estándar de producto)."

Para no perderse en este amplio tema de los estándares, se toma en cuenta los siguientes tipos como una clasificación:

- Estándares de organización
- Estándares de mercado (ej. Microsoft Windows, VHS)
- Estándares profesionales (ej. IEEE)
- Estándares industriales (ej. CD-V video-disco)
- Estándares internacionales (ej. ISO,IEC,ITU)

Los estándares representan acuerdos por consenso, por lo tanto no siempre son ideales. Su mayor valor consiste en la difusión de terminología, procedimientos, modelos y puntos de referencia.

Hoy en día existen varias organizaciones de estandarización internacional, algunas son regionales mientras que otras son globales. Las últimas están relacionadas con la ONU o son independientes, como por ejemplo la *International Telecomunication Union* (ITU).

La *International Electrotechnical Commission* (IEC) fue fundada en el año 1906 para definir estándares en eléctrica y electrónica , mientras que la *International Organization for Standarization* (ISO) fue creada en 1947 para abarcar otros temas. Ambas tienen por objetivo facilitar intercambio de bienes y servicios a nivel internacional.

En 1987, ISO e IEC decidieron formar el *Joint Technical Committe* (JTC), cuyo objetivo es elaborar estándares para la tecnología de información (IT).

En cuanto a la Ingeniería de Software, el estándar más antiguo, publicado en 1972 es el estándar militar de E.U., sobre el aseguramiento de calidad. Lo siguió en 1976 el estándar sobre los planes de aseguramiento de calidad de IEEE.

DEFINICIÓN DEL PROBLEMA

Actualmente, las empresas que tiene relación con el desarrollo de software, están solicitando de los egresados de las distintas carreras profesionales relacionadas con computación, conocimientos sobre aseguramiento de calidad en el desarrollo de productos de software. Los modelos de calidad existentes (ISO9000) son primordialmente para aplicaciones de ingeniería en general. Existe, sin embargo, esfuerzos por adaptar estos modelos (ISO9000-3, ISO15504, ISO9126) a empresas que desarrollan productos y servicios relacionados con computación, así como aplicaciones específicas de sistemas de información. Sin embargo, las empresas de desarrollo de software están buscando no solo la adecuación de sus procesos a dichos modelos de calidad, sino que además están buscando la evaluación de modelos de calidad exclusivos para el software, siendo actualmente el más importante, el Capability Madurity Model (CMM) (Modelo de Capacidad Madurez) desarrollado por el Software Engineering Institute (SEI) de la universidad Carnegi Mellon University (CMU) en los Estados Unidos.

Los modelos o sistemas de calidad mencionados indican lo que debe presentar una empresa para ser evaluada o certificada, pero no como lograr obtener las características solicitadas por dichos modelos o sistemas. Al revisar la literatura referente a dichos modelos o sistemas, mencionan que los aspectos que debe exhibir una organización para obtener la certificación o evaluación se pueden alcanzar de diversas maneras pero no menciona ninguna en particular.

Los modelos proponen desde obtener los niveles de calidad de manera binaria, es decir, está o no certificada la organización, hasta niveles discretos específicos, describiendo las diferentes áreas importantes para cada nivel, mencionando tan solo las metas, precondiciones y compromisos de la organización y sus integrantes. Por lo anterior, es una excelente área de oportunidad para desarrollar una investigación buscando de que manera se puede ayudar a las empresas a lograr el qué. Dentro de la propuestas que existen para lograr la mejora de una organización de desarrollo de software está el uso de las guías y estándares del IEEE. La base de datos está disponible en el UNAM por lo que existe la factibilidad de desarrollar el estudio y obtener resultados concretos y aplicables. Por otra parte, es necesario que los integrantes de las organizaciones posean la cultura, los conocimientos y la experiencia en el uso de los estándares para así poderlos adaptar a la organización y lograr así la evaluación o certificación del modelo o sistema en cuestión.

Actualmente, en la Facultad de Ingeniería, en las diferentes materias relacionadas con la computación, no se cubren aspectos y conceptos de aseguramiento de calidad en el desarrollo de software para las diferentes carreras que tiene relación con computación.

En la materia de Ingeniería en Programación, un tema del temario tiene que ver con calidad, sin embargo, no se puede revisar lo que ello implica a profundidad. Por otra parte, se puede cursar como optativa la materia de Calidad, siendo ésta de manera genérica para todas las ingenierías, la cual sería muy útil como requisito para materias relacionadas con la calidad y el aseguramiento de la calidad pero específicamente para productos de software. En esta materia podría hacerse el esfuerzo de identificar los estándares del IEEE que apoyan cada uno los momentos importantes en el desarrollo de software. Sin embargo, el tiempo no es suficiente, por lo que desarrollar un trabajo de tesis que sea referencia para otras materias y para esta misma materia en otros semestres, así como para consulta de interesados que laboran ya en empresas de desarrollo podría ser otro resultado importante del trabajo de investigación.

El objetivo es evaluar la posibilidad de usar y adoptar los estándares y guías del IEEE para apoyar a las organizaciones en el Proceso de Mejora de Desarrollo de Software (PMDS) y lograr así las características impuestas y descritas en los documentos de los modelos referentes a CMM o sistemas de calidad tipo ISO9000-2000.

MÉTODO

Para este trabajo se propone la revisión de los modelos de calidad CMM e ISO 9000-200 en el desarrollo de software para posteriormente estudiar los estándares y guías del IEEE e identificar la forma en que se pueden adoptar para apoyar a las organizaciones de desarrollo de software para obtener ya sea la certificación o la evaluación correspondiente.

INVENTARIO DE MATERIAS

Las materias requisito son todas las materias que estén relacionadas con la carrera de computación siendo éstas: Computación, Estructura de Datos, Estructura de Datos Discretas, Compiladores, Ingeniería en Programación, Calidad, Bases de Datos.

RESULTADOS ESPERADOS

Revisión de los modelos CMM e ISO 9000-2000 de calidad para organizaciones que desarrollen productos de software, revisión de los estándares y guías del IEEE relacionadas con dichos modelos, finalmente, la adecuación de dichos estándares y guías en el PMDS.

CAPÍTULO 1

GUÍAS PARA MEJORA DEL PROCESO DE SOFTWARE

La serie de Normas ISO 9000 son un conjunto de enunciados que especifican los elementos que deben integrar el *Sistema de la Calidad* de una empresa y como deben funcionar en conjunto estos elementos para asegurar la calidad de los bienes y servicios.

Las Normas ISO 9000 son generadas por la *Organización Internacional de Normalización,* cuya sigla es **ISO**. Esta organización internacional está formada por los organismos de normalización de casi todos los países del mundo. Los organismos de normalización de cada país producen normas que se obtienen por consenso en reuniones donde asisten representantes de la industria y de organismos estatales. De la misma manera, las Normas ISO se obtienen por consenso entre los representantes de los organismos de normalización enviados por cada país.

Ahora bien, un sistema es un conjunto de elementos que están relacionados entre sí. Es decir, hablamos de sistema, no cuando tenemos un grupo de elementos que están juntos, sino cuando además están relacionados entre sí, trabajando todos en equipo. Es por eso que un Sistema de la Calidad significa disponer de una serie de elementos como Manual de la Calidad, Equipos de Medición, Carpetas de Procedimientos, Personal Capacitado, etc., todo funcionando en equipo para producir bienes y servicios de la calidad requerida por los clientes. Los elementos de un sistema de la calidad deben estar *documentados por escrito*.

Es importante comprender bien esta definición para que se entienda que las Normas ISO 9000 no definen como debe ser el sistema de calidad de una empresa, sino que fija requisitos mínimos que deben cumplir los sistemas de la calidad. Dentro de estos requisitos hay una amplia gama de posibilidades que permite a cada empresa definir su propio sistema de la calidad, de acuerdo con sus características particulares.

Los cambios en las normas ISO 9000:2000, fueron muy representativos en cuanto a los principios básicos de la Gestión de la Calidad. Una vez que surge la idea de llevar a cabo todo un proceso de trabajo que conllevara a la certificación internacional, es necesario enfocarse primeramente en los principios que rigen la norma ISO 9001, ya que son considerados como la base de todo un proceso de cambios. Los requisitos de la norma ISO 9000:2000 son flexibles y algunos de ellos se pueden omitir dependiendo de las necesidades o características de cada organización.

La experiencia acumulada por la implementación de las normas ISO 9000 en cientos de miles de organizaciones en todo el mundo indican la necesidad de mejorarlas, hacerlas más amigables sobre todo para la pequeña y mediana empresa. Dicha experiencia ha mostrado que los resultados deseados se alcancen más eficientemente cuando las actividades y los recursos relacionados se gestionan como un proceso. En consecuencia uno de los caminos para lograr la mejora fue adoptar un sistema de gestión con un enfoque de procesos para lo cual se requirió desarrollar un modelo.

Este modelo unido a los ocho principios de la Gestión de la Calidad constituyen la parte medular del sistema o proceso de implantación de para la mejora continua. Estos principios de calidad son los siguientes:

1.- Organización enfocada al cliente. Las organizaciones dependen de sus clientes y por lo tanto comprender sus necesidades presentes y futuras, cumplir con sus requisitos y esforzarse en exceder sus expectativas.

2.- Liderazgo. Los líderes establecen la unidad de propósito y dirección de la organización. Ellos deben crear y mantener un ambiente interno, en el cual el personal pueda llegar a involucrarse totalmente para lograr los objetivos de la organización.

3.- Participación de todo el personal. El personal, con independencia del nivel de al organización en el que se encuentre, es la esencia de la organización y su total implicación posibilita que sus capacidades sean usadas para el beneficio de la organización.

4.- Enfoque a procesos. Los resultados deseados se alcanzan más eficientemente cuando los recursos y las actividades relacionadas se gestionan como un proceso.

5.- Enfoque del sistema hacia la gestión. Identificar, entender y gestionar un sistema de procesos interrelacionados para un objeto dado, mejora la eficiencia y la eficiencia de una organización.

6.- La mejora continua. La mejora continua debería ser el objetivo permanente de la organización.

7.- Enfoque objetivo hacia la toma de decisiones. Las decisiones efectivas se basan en el análisis de datos y en la información.

8.- Relaciones mutuamente benéficas con el proveedor. Una organización y sus proveedores son independientes y una relación mutuamente benéfica intensifica la capacidad de ambos para crear valor.

Los requisitos de la Norma ISO 9000:2000 son utilizados por las empresas a certificarse, desde que comienza la implantación del Sistema de Gestión de Calidad que más convenga a la empresa, hasta la evaluación en las auditorias finales.

Un sistema de calidad debe cumplir una serie de exigencias para que sea efectivo, pero dentro de estas exigencias debe hacerse una diferenciación muy clara entre los requisitos del producto y los requisitos del sistema de calidad.
Los requisitos para los productos pueden ser especificados por los clientes, por la propia organización o bien por la autoridad.

Los requisitos para los productos y en algunos casos para los procesos asociados pueden estar contenidos en especificaciones técnicas, normas de producto, normas de proceso o requisitos reglamentarios. Los requisitos del sistema de Gestión de la Calidad son complementarios a los requisitos del producto y se especifican en la norma ISO 9001:2000, son genéricos y aplicables a organizaciones de cualquier sector económico e industrial con independencia del producto que suministren y además hacen énfasis en el uso y aplicaciones técnicas del producto.

Antes de tratar las guías para poder lograr una certificación, es importante que se mencione a todos los estándares relacionados con el *proceso software*:

Definición de procesos estándar
- ISO 9000
- *European Software Agency* (ESA) PSS-05
- ISO 12207; IEEE 1074

Definición de un método de evaluación del proceso
- SEI's CMM
- *European Bootstrap Method*
- ISO 15504 (SPICE)

Definición de métodos de mejora del proceso
- *Quality Improvement Paradigm* (QIP)
- *Personal Software Process* (PSP)
- Gestión de la Calidad Total

La calidad del software puede medirse después de elaborado el producto, pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del *software*.

Cuando se trabaja para construir un producto o un sistema, es importante seguir una serie de pasos predecibles, como un mapa de carreteras que ayuda a obtener el resultado oportuno de calidad. El mapa de carreteras a seguir es llamado “proceso del software”.

Los ingenieros de software y sus gestores adaptan el proceso a sus necesidades y entonces lo siguen.

El proceso es importante porque proporciona estabilidad, control y organización a una actividad que puede, si no se controla, ser caótica. El proceso de software es un marco de trabajo de las tareas que se requieren para construir software de alta calidad.

La calidad esta relacionada directamente con la Ingeniería de Software utilizada en el proceso de desarrollo.

Proceso

Cuando se presenta un servicio o se crea un producto, siempre se sigue una secuencia de pasos para lograr una serie de tareas.
Podemos pensar al conjunto ordenado de tareas como un proceso; una serie de pasos que involucran actividades, restricciones y recursos que producen una determinada salida esperada.

Un proceso involucra un conjunto de herramientas y técnicas. Cualquier proceso tiene las siguientes características:

- El proceso utiliza recursos, está sujeto a una serie de restricciones y genera productos intermedios y finales.
- El proceso puede estar compuesto por subprocesos que se encadenan de alguna manera
- Cada actividad del proceso tiene criterios de entrada y salid, de modo que se conoce el inicio y final de una actividad.
- Las actividades se organizan en una secuencia de modo que resulta claro cuando una actividad se realiza en orden relativo a otras actividades
- Todo proceso tiene un conjunto de principios orientados que explican las metas de cada calidad
- Las restricciones o controles pueden ser de aplicación a una actividad, recurso o producto.

Los proceso son importantes porque imponen consistencia y estructura sobre un conjunto de actividades. Así entonces, se concluye que un proceso es un conjunto de procedimientos organizado de tal modo que los productos se construyen para satisfacer un conjunto de metas o estándares.

Para la mejora del proceso de software se deben tomar en cuenta ciertos aspectos, en este caso son una serie de guías para mejorar el proceso. Esto se presenta en una secuencia de 12 pasos que son los siguientes:

1. Búsqueda de las regulaciones aplicables

Se debe buscar cuidadosamente todas las regulaciones gubernamentales y los estándares de calidad para la industria que más podrían satisfacer para estas mejoras del proceso.
También se deben identificar las discrepancias entre el actual proceso de la organización y lo estipulado en la regulación aplicable.

2. Compilar datos de proyectos previos

Este punto es importante, pues la información de proyectos previos puede proporcionar ayuda invaluable en planear las mejoras para el proceso de software. Mucha de esta información se avala por largos periodos de tiempo en que ya se tengan en uso.
Además, información importante puede ser avalada tomando en cuanta las siguientes fuentes:

- Comparaciones entre lo planeado y horarios del proyecto reales
- Costos actuales del proyecto
- Reportes de pruebas
- Reportes de problemas
- Software de métricas
- Existencia de documentos (de proyectos previos)

3. Plan del alcance para las mejoras de proceso

Se debe tomar como una prioridad enfocarnos en corregir y regular los problemas en proyectos previos. Así podemos usar las técnicas y herramientas nuevas para la mejora y la eficiencia de en la lista de problemas antes mencionados.
Los estándares de la IEEE que especifican el formato y contenido de la documentación principal que son hechos durante un proyecto de software son el estándar 830 (Especificación de documentos) y el estándar 829 (Documentación de pruebas del software).

4. Obtener el compromiso de administración

Siempre hay riesgos asociados con los cambios en algún proceso y por eso la administración debe estar preparada para posibles reacciones de la acciones llevadas acabo.

Por lo anterior podemos tomar en cuenta ciertos conceptos llave al implementar el plan de mejora, los cuales pueden ser los siguientes:

- Anticipar los beneficios
- Descripción de los cambios
- Estrategia en las fases
- Evaluación del programa

5. Construir el soporte de la conexión de nivel

Anteriormente se debió obtener soporte administrativo para la mejora del proceso, por lo que se debe también tener un consenso o junta entre el personal afectado. De hecho, en algunos casos, el responsable debe promover el soporte para la mayoría del personal que vaya a ejecutar las mejoras del proceso. Deberán de tomarse en cuenta sugerencias para éstas mejoras, pero deberán solicitarse sistemáticamente. Además, los participantes deberán tener oportunidad de exponer su opinión al respecto y se sugiere una presentación para exponer las razones por la cuales se procede a realizar cambios en el proceso.

6. Plan de los procedimientos de operación de los estándares

Los procedimientos de operación del estándar (SOP's) deberán describir el desarrollo de software y los procesos de validación. Los SOP's deberán especificar terminología, modelos de proceso, documentos que serán creados, técnicas que se usarán y los estándares correspondientes. El SOP podría deberá ser correcto, inequívoco, completo, consistente y verificable. Diagramas, formas y otras conexiones pueden ser muy útiles en los SOP's. Cuando un nuevo personal entra a la organización, los SOP's pueden ser una herramienta importante para enseñar las responsabilidades al nuevo miembro del equipo.

7. Conducir una revisión de equipo

Cuando el plan de SOP's está listo, deberán ser revisados por el equipo evaluador. El estándar 1028 es una excelente guía para hacer revisiones adecuadas.

8. Aprobación y control de procedimientos de operación de los estándares

Los SOP's deberán ser tratados como un documentado controlado en la organización, la manera en que el documento será controlado dependerá ya de la organización. Es recomendado que se mantenga un archivo con las versiones pasadas de los SOP's, ayudarán en la planeación de futuras mejoras de proceso.

Uno de los aspectos más críticos de la propiedad de control en los SOP's, es la distribución de estos al personal que los debe usar.

9. Especificar las fases para las mejoras de proceso

Se podrá restringir las fases en las que serán revisadas las SOP's del proyecto piloto, antes de su uso.

10. Entrenar personal en mejoras de los proceso

Antes de usar un nuevo SOP's, se debe capacitar a todo el personal en el uso de este.

11. Usar superviso en las mejoras de proceso

Al usar los nuevos SOP's, estos deberán ser supervisados desde el punto de vista de la aplicación y la medición. Al comenzar a usarlo deberá ser verificada cada fase del proyecto según sea completada.

12. Evaluar los éxitos de las mejoras de proceso

Establecer un efectivo programa de mejora del proceso, pues se deberán evaluar los éxitos y fracasos de dicha mejora. Esto servirá como retroalimentación al programa de mejora para poder realizarle futuros cambios y continuar mejorando el desarrollo de software y la validación de procesos.

CAPÍTULO 2

PANORAMA SOBRE ESTÁNDARES EN INGENIERÍA DE SOFTWARE

Los estándares definen los procesos a través los cuales el software es desarrollado y validado. No sólo definen la documentación a ser creada, sino cómo estos documentos deben ser evaluados y aprobados.

Los beneficios de implementar los estándares son:

a) Incremento en la calidad del software ya que se reduce el número de defectos y por lo tanto bajan los costos de mantenimiento.
b) Reduce el tiempo y costos del proyecto ya que ayudan a detectar los errores en las fases tempranas del proyecto, como son durante los requerimientos de análisis y diseño de fases.
c) Cumplir con la normatividad vigente ya que ayuda a satisfacer las regulaciones gubernamentales y estándares de calidad lo cual es esencial en auditorias y certificaciones.
d) Coordinación optimizada de proyectos, lo cual implica una mejor planificación y planeación de los proyectos.

Así también, la calidad es definida por IEEE como el grado al cual un sistema, componente o proceso conoce los requerimientos especificados y conoce las necesidades del cliente ó expectativas (es la satisfacción total del cliente).

Un sistema de calidad debe consistir de las herramientas más modernas para asegurar el ciclo de vida del software, esto es porque la evaluación de los sistemas es tan compleja que nunca puede ser tan exhaustiva.

Mejorar la calidad del software tiene ventajas administrativas por las siguientes razones:

- Incremento de ventas
- Disminución de los costos
- Disminución de efectos
- Reutilización del software

Sin un proceso modelo y sin estándares, el desarrollo de un proyecto de software puede llegar a convertirse en una caja negra. Además, a las empresas les interesa un sistema de calidad que sea efectivo, pues si lo es, reduce tanto los costos como el tiempo.

Como ya se señaló, los estándares ISO-9000 requieren de procesos sistemáticos para el desarrollo y validación del software. La certificación por ISO-9000 es necesaria si se quiere entrar a un mercado internacional (sobre todo en Europa y EUA), pero estos estándares no especifican como deben ser validados los software así que cada empresa puede desarrollar sus modelos a seguir, los cuales deben estar sustentados por estándares internos propios de la organización.

Los estándares también permiten la detección temprana de fallas en el ciclo de vida del software. Las actividades de verificación durante las fases tempranas pueden identificar problemas sistemáticos de calidad, permitiendo acciones correctivas de manera oportuna.

Los estándares de la IEEE tiene una estructura y se comenzará por describir la estructura del modelo más básico, que es la estructura del modelo del SESC (Comité de Estándares de Ingeniería del Software de la IEEE), seguido de una organización alternativa.

La aplicación de cada estándar individualmente es analizada en función de que tan critica sea su aplicación para el software.

2.1 Estructura del SESC.

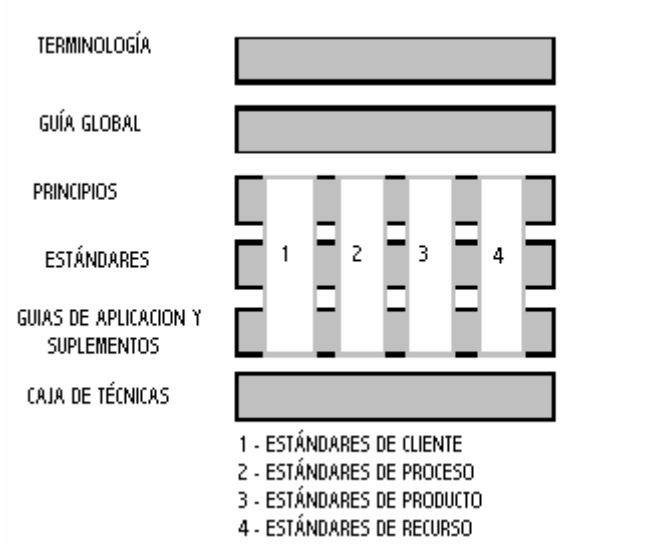
El SESC creo una estructura para describir la colección de estándares, que está compuesta de 6 capas que son las siguientes:

1. *La terminología*: son los documentos que describen los términos y el vocabulario.
2. *Guía Global*: es un documento que describe de manera general la coleccione entera.
3. *Principios:* son documentos que describen los objetivos elementales.
4. *Estándares fundamentales*: estándares que son básicos para la conformidad.
5. Guías de Aplicación y Suplementos: guías y suplementos que dan consejo para el uso de los estándares en varias situaciones.
6. *Caja de Herramientas o Técnicas:* son la descripción de técnicas que pueden ser útiles para ayudarte a implementar el mas alto nivel en tus documentos.

La clasificación de los objetos de este modelo podría ser de la siguiente manera:

- *Estándares para el cliente*.- describen la interacción entre el cliente y el proveedor del software.

- *Estándar de proceso.-* describe el ciclo de vida del producto o servicio incluyendo desde la adquisición, suministro, el desarrollo, mantenimiento y operaciones.
- *Estándar del producto.-* estos estándares te dicen los requerimientos de cada productos (características, especificaciones, etc).
- *Estándares de la fuente.-* estos te recomiendan la documentación apropiada, los métodos, los modelos y las herramientas para el buen manejo de un software.



Estructura del modelo del SESC

Existe otra organización a la cual se le puede decir que es una organización *alternativa* del SESC para los estándares, y en ella hay 4 categorías que son:

- Terminología
- Procesos del ciclo de vida
- Herramientas
- Reuso

Así también, el manejo del proyecto, la planeación, la documentación y las medidas forman este modelo alternativo.

Los actuales estándares IEEE no fueron desarrollados de acuerdo con la estructura del SESC. El SESC fue desarrollado después para darle una clasificación a los estándares y obviamente por eso no siempre encajan todos los estándares en esta estructura.

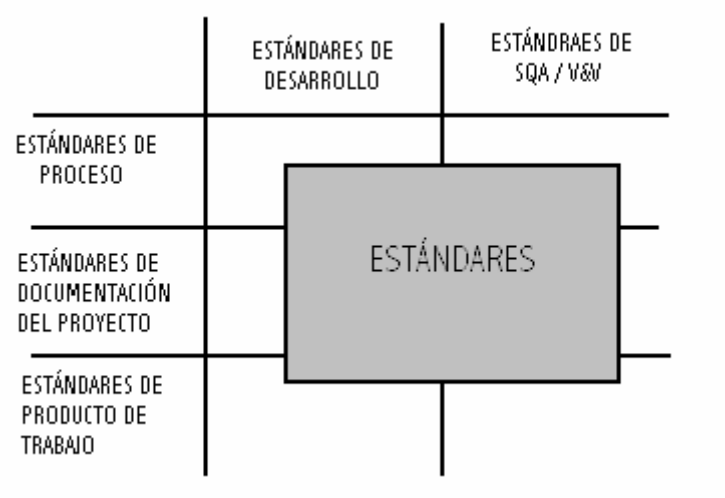
2.2 Modelo organizacional simplificado.

Existe también otro modelo, llamado modelo organizacional simplificado, el cual se restringe también a una simple población de estándares. En la siguiente figura lo podemos ver y el modelo se describe como sigue.

- *Proceso*. Estándares, descripciones y otros medios por los cuales una organización describe el proceso a ser usado.

- *El proyecto*. El proyecto del software es una entidad compleja consistente de gente trabajando por un periodo prolongado de tiempo usando herramientas y otros recursos para crear un producto. Los planes y actividades que se llevan a cabo en el proyecto, forman parte de los elementos del proyecto.

- *El producto del trabajo.* Éste se refiere al software como tal (el producto), los documentos que son generados durante el proyecto (especificaciones, descripciones y en general documentos del software) son los que forman parte de esta categoría.



Clasificación simplificada de los estándares

Posteriormente, se dividen las actividades de ingeniería del software en dos categorías:

1. El desarrollo
2. La verificación y la validación

2.3 Aplicación de los estándares.

Según “Sommerville”, los estándares son útiles porque agrupan lo mejor y más apropiado de las buenas prácticas y usos del desarrollo de software; engloban los “conocimientos” que son patrimonio de una organización; proporcionan un marco para implementar procedimientos de aseguramiento de la calidad y por último proporcionan continuidad entre el trabajo de distintas personas.

Hay una obvia y directa relación entre los estándares de le IEEE y el modelo de madurez. Este puede ser usado según la madurez de la organización y para el plan de mejoras del proceso de software.

Cuando el nivel critico de un una aplicación aumenta , también puede aumentar el número de estándares que se debe usar en el nivel al que corresponda. Lo crítico de la aplicación esta en función de la probabilidad de riesgo y error.

Se debe analizar cuidadosamente que leyes, regulaciones, guías y otros estándares deben aplicarse para que un proceso sea aceptado. La implementación de estándares está basado en lo básico de la aplicación, el tamaño del equipo del proyecto, la madurez de la organización y cualquier requerimiento externo par el proceso.

2.4 CMM (Modelo de Madurez de Capacidad)

En los últimos años se ha hecho mucho énfasis en la “madurez del proceso“. El Instituto de Ingeniería de Software (SEI) ha desarrollado un modelo completo que se basa en un conjunto de funciones de ingeniería de software que deberían estar presentes conforme organizaciones alcanzan diferentes niveles de madurez del proceso.

CMM es un modelo de procesos, o una colección estructurada de elementos que describen características o procesos probados por la experiencia de ser efectivos. Te dice *que hacer*, no *como hacerlo*.

El enfoque del SEI proporciona una medida de la efectividad global de las prácticas de ingeniería del software de una compañía y establece cinco niveles de madurez del proceso, que se ejemplifican en la siguiente tabla y que se definen de la forma siguiente:

Nivel crítico de la aplicación	*Tamaño del equipo*	*Estándares*	*Madurez del proceso*	*Requerimientos de proceso externos*
Ninguno	Individual	**0** 12207.0 (SLCP)	Ninguno	Ninguno
Bajo	Muy pequeño	**1** 830 (SRS), 829 (Test Doc.)	Primer esfuerzo para formalizar	Bajo
Moderado	Pequeño	**2** 1016 (SDD), 1058 (SPMP), 1028 (SW Reviews)	Segunda fase de formalización	Moderado
Mayor	Medio	**3** 730 (SQAP), 1012 (SW V&V), 828 (SCMP)	Tercera fase	Mayor
Alto	Grande	**4** 1074 (SLCP), 1061 (QMM), 982.1 (Measures)	Maduro	Alto
Alto	Muy Grande	**5** Otros estándares	Alto	Alto

Tabla de aplicación de estándares en un modelo de madurez simple

Nivel 0.

Es el modelo de madurez que corresponde a la adopción del proceso del mas alto nivel y este nivel es apropiado solamente si no hay absolutamente nada crítico en el software y no hay requerimientos externos para estándares adicionales.

La 12207 da un apropiada guía de madurez del nivel cero y establece:

1. Proceso primarios que incluyen la adquisición y suministro, el desarrollo, la operación y el mantenimiento.
2. Los proceso de soporte que incluyen la documentación, configuración, aseguramiento de calidad, verificación, validación, la auditoria y problemas de resolución.
3. Procesos organizacionales, el manejo, la infraestructura, mejoramiento y entrenamiento. Esta guía ayuda a resolver los problemas que se me puedan presenta durante el desarrollo del software.

Este estándar no especifica muchas de las tareas que deben llevarse a cabo en este proceso, por lo que debe apoyarse de otras estándares para conocer formatos y estructuras de los planes o de proceso.

Nivel 1: Inicial.

El éxito depende de esfuerzos heroicos y personales más que de procesos adecuadamente definidos. No hay proceso definido implícita o explícitamente.

El nivel uno es el modelo de madurez que corresponde al mas simple proceso que pertenece a la caja negra del nivel cero. Este nivel es solamente adecuado para aplicaciones de bajo riesgo y bajos requerimientos externos para el proceso.

La Especificación de Requerimientos del Software es un documento que podría mostrar lo que hace el software y la Documentación de Prueba es demostrada para demostrar que el software actualmente “hace eso”.

Para este nivel se aplica el estándar 829 de la IEEE (Especificación de Requerimientos del Software), que define un desarrollo de especificaciones completo e inequívoco de las características funciónale del software. El SRS es probablemente el documento de software más importante. En este nivel el proceso de madurez esta restringida al nivel del sistema debido a que las unidades del diseño del software no están especificadas.

Nivel 2: Repetible.

Se establecen políticas y procedimientos para llevar a cabo un proyecto. Una función de calidad asegura que se cumplen dichos procedimientos. Se obtienen niveles de calidad parecidos a proyectos anteriores.

El nivel 2 tiene un modelo de madurez que formaliza varias partes importantes que pertenecen al nivel 1. El nivel 2 formaliza la creación de un plan, la documentación del diseño del software y un acercamiento sistemático a la revisión de los productos. Este nivel es adecuado para proyectos de riesgo moderado y es recomendado para una segunda fase de formalización del proceso del software. Los estándares de la IEEE que se aplican en este nivel son el 1058 (Plan para la Administración del Proyecto de Software), 1016 (Descripción del Diseño de Software) y 1028 (Revisiones del Software).

Nivel 3: Definido.

Se adopta un proceso software estándar y se adapta a cada proyecto.

Este modelo de madurez abarca los estándares del nivel 2 y además implican proyectos adicionales tal como la configuración, evolución, aprobación o desaprobación e implementación de cambios a productos controlados. El nivel del proceso es adecuado para aplicaciones de mayor riesgo. Este nivel es recomendado para la tercera fase de formalización del proceso. Los estándares de al IEEE aplicados en este nivel son el 730(Planes de Aseguramiento de Calidad

del Software), el 1012 (Verificación y Validación), y el 828 (Planes de la Administración de Configuración del Software).
En este nivel también se requiere un mínimo de documentación que incluye requerimientos de especificación y descripción del diseño.

Nivel 4: Gestionado.

La calidad del producto y del proceso es medida, predecible y cuantificable. Se pueden usar dichas medidas ("métricas del software") para detectar situaciones excepcionales y corregirlas.

El nivel 4 en el modelo de madurez representa la conformidad total con todos los estándares de la IEEE, y corresponde a un proceso de madurez que es adecuada a las mas altas aplicaciones de riesgo del software, un nivel crítico de aplicación muy alto. Los estándares que se incluyen en este nivel son el 1074, encargado de desarrollar el Proceso del Ciclo de Vida del Software (SCLP); el 1061 (Metodología de Métricas de Calidad para el Software); y el 982.1 (Diccionario de medidas para un producir un software confiable).

Nivel 5: Optimizado.

El proceso es continuamente mejorado usando las medidas obtenidas de procesos anteriores.

Corresponde al mas alto nivel del proceso de madurez aun para las mas grandes organizaciones. Debido a que los estándares IEEE son continuamente revisados, nuevos estándares son adicionados y continuamente el programa es actualizado, el nivel 5 esta fuera de la investigación de esta tesis.

El SEI ha asociado áreas claves del proceso a cada uno de los niveles de madurez (Key Process Areas: KPA). Las KPA's describen esas funciones de la ingeniería del software que se deben presentar para satisfacer una buena práctica a un nivel en particular. Se definen diferentes KPA's y se distribuyen en diferentes niveles de madurez del proceso como se muestra a continuación:

Nivel 1. No aplican

Nivel 2. Administración de requerimientos
Aseguramiento de calidad de software
Administración de configuración de software
Administración de subcontratación
Planeación de proyectos de software
Seguimiento de proyectos de software

Nivel 3. Enfoque organizacional en procesos
Definición organizacional de procesos
Programas de entrenamiento
Administración de integración de software

Ingeniería de software de producto
Coordinación intergrupal
Política de revisiones

Nivel 4. Administración de calidad de software
Administración cuantitativa de procesos

Nivel 5. Prevención de defectos
Administración de cambios tecnológicos
Administración de cambios de procesos

Actualmente existen dos modelos de madurez del SEI (Software Engineering Institute) que han tenido más aceptación para las organizaciones y áreas que se dedican al desarrollo del software y que últimamente han generado gran controversia en la comunidad mundial: SW-CMM y CMMI.

Como consecuencia del estudio de estas dificultades y de la preparación de la siguiente generación de sus modelos de madurez, en diciembre de 2001 el Instituto de Ingeniería de Software (SEI) publicó la versión 1.1 del CMMI-SE/SW/IPPD (Modelo de Madurez de Capacidad Integrado para varias disciplinas). Este modelo, tiene el propósito de proporcionar una única guía unificada para la mejora de múltiples disciplinas tales como ingeniería de sistemas (SE), ingeniería del software (SW) y el desarrollo integrado del producto y del proceso (IPPD). Más recientemente, el esfuerzo está siendo ampliado para incluir requisitos específicos para la gestión y control de proveedores. Además, debido a la existencia de un modelo internacional para la mejora de los procesos del software y determinación y evaluación de su capacidad (ISO/IEC TR 15504), hay un compromiso de que el CMMI tenga conformidad y compatibilidad con dicho modelo internacional.

Los niveles de madurez del CMMI tienen básicamente las mismas definiciones que sus modelos anteriores, aunque se ha modificado la terminología de algunos de los niveles. Los niveles 1, 3 y 5 conservan sus nombres originales (Inicial, Definido y Optimizado), mientras que los niveles 2 y 4 ahora se denominan Gestionado y Gestionado de forma cuantitativa, respectivamente, con la intención de acentuar la evolución de los procesos de gestión a un enfoque de toma de decisiones basadas en datos y en hechos.

El CMMI esta caracterizado por 28 áreas de proceso para las cuatro disciplinas que cubre actualmente, es decir, Ingeniería de sistemas (SE), Ingeniería del software (SW), Desarrollo integrado del producto y del proceso (IPPD) y la gestión de compras y control de proveedores (A). Aunque muchas de las áreas de proceso (PA) definidas en el CMMI tengan los mismos nombres que las áreas clave de proceso (KPA), definidas en su modelo anterior el SW-CMM, existen una serie de cambios significativos en cuanto al enfoque y el alcance de sus actividades y objetivos.

Existen varias representaciones del CMMI atendiendo a las diversas necesidades de las organizaciones que quieren realizar la mejora de sus procesos. La representación organizada hace especial énfasis en el grado de madurez de los procesos (a semejanza del SW-CMM), mientras que la representación continua hace hincapié en la capacidad de ciertas áreas para realizar adecuadamente sus actividades.

Podemos resumir las mejoras que aporta el nuevo modelo de la siguiente manera:

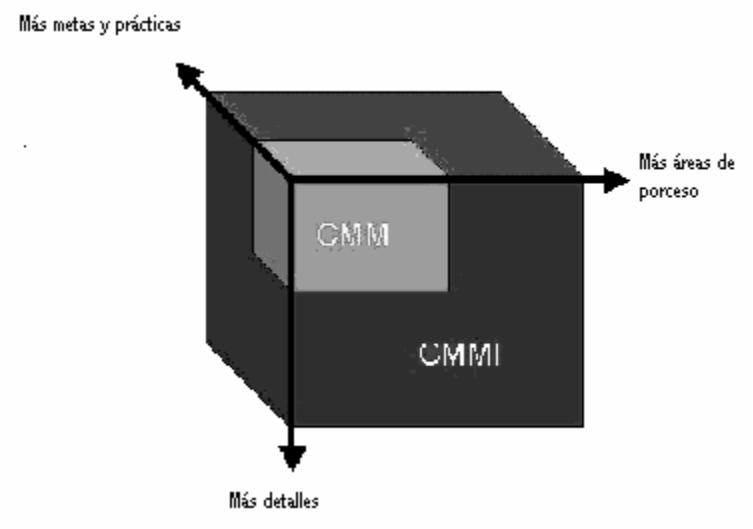
- Integra cuatro modelos que tienen una serie de diferencias de origen.
- Desarrolla un marco de actuación para permitir el crecimiento de otras disciplinas.
- Se han agregado las áreas de proceso para hacer un mayor énfasis sobre algunas prácticas importantes:

Medición y análisis (nivel 2)
Gestión de riesgos (nivel 3)
Análisis sistemático y puesta en práctica de las decisiones acordadas (nivel 3)
Todas las nuevas áreas de proceso definidas en la categoría de Ingeniería
Gestión integrada (nivel 3) y cuantitativa (nivel 4) de las compras de la organización.

- Nuevo énfasis sobre el producto, así como sobre los procesos, incluyendo las interacciones con el cliente.
- Mayor importancia, desde las fases iniciales, del análisis y la medición de los procesos empresariales.
- Cobertura de servicios, así como de sistemas.
- Especial énfasis sobre la capacidad de los procesos y madurez de la organización en su conjunto (no exclusivamente en el área de ingeniería del software).
- Mejor cobertura de la gestión de ingeniería integrada.
- Énfasis sobre las mejoras medibles y cuantificables para alcanzar los objetivos del negocio empresarial.

Control de Calidad objetivo e incorporado a cada uno de los procesos.

- Existe un nuevo enfoque de la formación. La educación y el entrenamiento adecuado para la mejora de la eficacia y de la eficiencia.
- Favorece el establecimiento de un ambiente adecuado para la gestión de los cambios dentro de la organización.
- Proporciona consistencia y compatibilidad con la norma ISO/IEC TR 15504.
- Proporciona compatibilidad con los principios, requisitos y recomendaciones de la nueva norma ISO9000:2000.
- Sienta las bases para que las organizaciones del sector de desarrollo del software se encaminen hacia el ciclo de la mejora continua.


Más metas y prácticas
Más áreas de
porceso
CMM
CMMI
Más detalles

CAPÍTULO 3

EL CICLO DE DESARROLLO DEL SOFTWARE

Es importante dar dos definiciones establecidas de lo que es el ciclo de vida del software, como son:

"Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software"

IEEE 1074

"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 los requisitos hasta la finalización de su uso"

ISO 12207- 1

Cuando el proceso implica la construcción de algún producto, se refiere al proceso como un ciclo de vida. Entonces el proceso de desarrollo del software suele denominarse ciclo de vida del software, porque describe la vida de un producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento.

El *Ciclo de Vida de Desarrollo de Software* se refiere a los procesos primarios, tal como adquisición del software, abastecimiento, desarrollo, operación, mantenimiento, así también como el soporte y organización. Para proyectos individuales el estándar 12207 puede ser adaptado junto con el *modelo cíclico de vida del software* (SLCM).

Adaptación significa la eliminación de procesos no aplicables y el establecimiento de una secuencia de actividades especificadas en el estándar 12207 de acuerdo al modelo SLCM seleccionado.

El estándar 1074 es el estándar que se encarga del desarrollo del proceso del ciclo de vida del software y describe como es que durante el diseño del software se selecciona un SLCM, como se combina este con el proceso organizacional y crean el proceso cíclico de vida del software.

El estándar 1074 describe su propio esquema de clasificación y actividades y la selección de un SLCM.

3.1 Estándar 12207.0 de la IEEE: "Principios"

La colección IEEE ha quedado armonizadas con los estándares internacionales así como regionales.
Esta guía ha descrito las actividades a ser llevadas a través de 5 ciclos de vidas primarios, 8 de soporte y 4 presos organizacionales.

Cada proceso es dividido en un set de actividades y cada actividad es descrita en términos de una tarea específica requerida. Los procesos de ciclos de vida que define este estándar son los siguientes:

1. *Adquisición de procesos.* Define las actividades del cliente, la organización, el software y servicio del software.
2. *Abastecimiento de proceso.* Define las actividades del que abastece, la organización, el software y el servicio del software para el que lo adquiere.
3. *Desarrollo del proceso.* Define las actividades del que desarrolla, la organización que lo define y desarrollo del producto del software.
4. *Proceso de operación.* Define las actividades del operador, la organización que otorga el servicio de operación de un sistema de una computadora y el ambiente para los usuarios.
5. *Mantenimiento del proceso.* Define las actividades del mantenimiento. Este proceso la instalación y el retiro del software.

Se tiene que el más complejo de estos procesos es el desarrollo de procesos, el cual está descrito con tres actividades:

1. Implementación del proceso
2. Análisis de requerimientos del sistema
3. Diseño de la arquitectura del sistema
4. Análisis de requerimientos del software
5. Diseño de la arquitectura del software
6. Diseño detallado del software
7. Codificación y prueba del software
8. Integración del software
9. Prueba de cuantificación del software
10. Integración del sistema
11. Prueba de cuantificación del sistema
12. Instalación del software
13. Aceptación del software

El estándar también define 8 procesos de soporte del ciclo de vida del software, y son los siguientes:

1. Proceso de documentación
2. Configuración del proceso administrativo
3. Proceso de aseguramiento de calidad
4. Proceso de verificación
5. Proceso de validación
6. Procesos de revisión
7. Proceso de auditoria
8. Proceso de resolución del problema

Y por último, el estándar define la organización de los procesos del ciclo de vida de software, que son los siguientes:

1. Procesos administrativos
2. Procesos de infraestructura
3. Procesos de mejora
4. Procesos de entrenamiento del personal

El estándar 12207 se define como un proceso para llevar acabo el entrenamiento básico de su estándar internacional para un proyecto de software.

El estándar internacional contiene un conjunto de procesos actividades y tareas diseñadas para hacer adaptadas a los proyectos de software.

Se tiene 2 estándares que se especifican para los procesos del ciclo de vida del software y actividades asociadas, y éstos son el estándar 1012 (Verificación y validación del software) y el 1074 (Desarrollo del proceso del ciclo de vida del software).

3.2 Estándar 1074 de la IEEE: "Desarrollo del Proceso del Ciclo de Vida del Software"

Este estándar va a ayudar a elegir el SLCM adecuado, registrar las actividades dentro del estándar SLCM y aplicar los procesos operacionales activos (OPA), lo cual será encaminado a crear el proceso de ciclo de vida del software (SLCP) para un proceso especifico.

El proceso de desarrollo del SLCP, en acuerdo con el estándar 1074, involucra 3 pasos:

- Selecciono mi SCLM
- Registras tus actividades dentro de SCLM
- Estableces el SLCP

Hacer un registro de actividades sobre el SLCM es establecer una secuencia de actividades de se este estándar (1074) acorde al SLCM seleccionado. Una vez que se realiza esto, se crea el SLC y entonces el SLC es

una secuencia de actividades específicas del proyecto que es creado al registrar las actividades del estándar 1074 sobre un SLCM seleccionado, o sea que el SLC que es más especifico, para cada empresa.

Las herramientas que te ayudan a definir la organización del proyecto del software son llamadas *procesos organizacionales activos* (OPA'S). Una vez que el SLC ha sido desarrollado, las OPA's adecuadas son aplicadas para establecer el SLCP.

Son muchos los modelos de proceso que se describen en la literatura sobre ingeniería de software. Como ya se sabe, un modelo de proceso es una estrategia de desarrollo que acompañe al proceso, métodos y capas de herramientas. En esta tesis se citan solo algunos modelos de proceso, que son los siguientes:

Modelo de cascada

Uno de los primeros modelos que han sido propuestos es el modelo de cascada, donde las etapas se representan cayendo en cascada, desde una etapa hacia la siguiente. Una etapa de desarrollo debe completarse antes de dar comienzo a la siguientes. De esta forma, cuando los requerimientos del cliente han sido identificados, analizados para comprobar su integridad y consistencia, y los requerimientos han sido documentados, entonces el equipo de desarrollo puede seguir con las actividades del diseño del sistema. El modelo en cascada presenta una visión muy clara de cómo se llevan a cabo las etapas durante el desarrollo.
El modelo en cascada ha sido utilizado para prescribir las actividades de desarrollo de software en una variedad de contextos.

El modelo en cascada puede ser muy útil, ayudando a los desarrolladores a hacer diagramas de las actividades a realizar. Su simplicidad hace que sea fácil explicarlo a los clientes que no están familiarizados con el desarrollo de software.

Modelo de prototipos

El prototipado puede por sí mismo ser la base de un modelo de proceso efectivo. Dado que el modelo de prototipos permite que todo el sistema , o algunas de sus partes, se construyan rápidamente para comprender o aclarar aspectos, tiene el mismo objetivo que un prototipo de ingeniería, donde los requerimientos o el diseño requieren la investigación repetida para asegurara que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución. Pueden eliminarse uno o más de los lazos para el v de los requerimientos, el diseño o el sistema, dependiendo de lasa metas del prototipado. Sin embargo, el objetivo es el mismo: reducir el riego y la incertidumbre en el desarrollo.

El diseño inicial es revisado hasta que los desarrolladores, los clientes, y los usuarios están satisfechos con el resultado. A veces el análisis de las alternativas de diseño revela un problema con los requerimientos, y los desarrolladores retornan a las actividades de requerimientos para reconsiderar y cambiar la especificación de éstos. Finalmente, el sistema se codifica y se analizan las

alternativas, con la posibilidad de repetir nuevamente las etapas de diseño y requerimientos.

Desarrollo por fases: incrementos e iterativos

Existen varias maneras para que los desarrolladores decidan como organizar el desarrollo en versiones. Los dos enfoques más populares son el desarrollo incremental y el desarrollo iterativo. En el desarrollo incremental, el sistema, tal como está especificado en los documentos de requerimientos, es particionado en subsistemas de acuerdo con su funcionalidad. Las versiones se definen comenzando con un subsistema funcional pequeño y agregando funcionalidad con cada nueva versión. El desarrollo incremental va construyendo gradualmente su funcionalidad completa con cada nueva versión. En cambio el desarrollo iterativo entrega un sistema completo desde el principio y luego cambia la funcionalidad de cada subsistema con cada nueva versión.

Utilizando el desarrollo iterativo debemos proporcionar lasa formas primitivas de los tres tipos de funcionalidad en la versión 1. Cada versión, de alguna manera, es una mejora sobre las precedentes.

En realidad, muchas organizaciones utilizan una combinación de desarrollo iterativo e incremental. Una nueva versión puede incluir nuevas funcionalidades, pero la funcionalidad que ya existe en la versión actual puede hacer sido mejorada.

El modelo en espiral

Boehm examinó el proceso de desarrollo del software a la luz de los riesgos involucrados, sugiriendo que un modelo en espiral podía combinar las actividades del desarrollo con la gestión del riesgo, para minimizar y controlar el riesgo. El modelo es espiral es en cierto sentido semejante al desarrollo iterativo.

Comenzando con los requerimientos y una plan inicial para el desarrollo, el proceso inserta un paso para evaluar riesgos y construir prototipos de las alternativas, antes de escribir el “concepto de las operaciones” en un documento que describe el más alto nivel cómo debe trabajar el sistema. Desde este documento se especifica un conjunto de requerimientos, que son minuciosamente inspeccionados para asegurar que sean tan completos y consistentes como sea posibles. Así, el concepto de operaciones es el producto de la primera iteración y los requerimientos son el principal producto de la segunda. En la tercera iteración el desarrollo del sistema produce el diseño, y en la cuarta habilita las pruebas.

Con cada iteración, el análisis de riegos pondera diferentes alternativas a la luz de los requerimientos y restricciones, y el prototipado verifica el grado de factibilidad o de deseo antes de seleccionar una alternativa en particular. Cuando se identifican riesgos, los gerentes de proyecto pueden decidir cómo eliminarlos o minimizarlos.

CAPÍTULO 4

“ESTÁNDARES DE TRABAJO Y PRODUCTO DE SOFTWARE”

4.1 Estándares de producto de trabajo

Hay estándares para la documentación de usuario, especificaciones y requerimientos, descripción del diseño del software y documentación de prueba.

En este conjunto de estándares, relacionados con el producto de trabajo, lo más difícil de entender es la diferencia entre la Especificación de Requerimientos del Software (SRS) y la Descripción del Diseño del Software. En realidad es muy simple de comprender esta gran diferencia: La Especificación de Requerimientos del Software dice “que” debe hacer el software y la Descripción del Diseño del Software dice “como” debe hacerse eso (como se debe obtener la meta, el objetivo fundamental, la serie de tareas que debe realizar el software).

La Especificación de Requerimientos del Software puede ser desarrollada y revisada por el cliente y el proveedor, por lo que, en esta documentación, no necesariamente debe estar involucrada en el desarrollo una persona técnica. A diferencia de éste, la Descripción del Diseño del Software es desarrollado por un profesional de software para otro profesional de software. Con estos dos aspectos se puedo comprender mejor la gran diferencia entre estos dos tipos de documentación, pero se debe tener muy claro que el SRS es una etapa inicial del proyecto que completa a la Descripción del Diseño.

4.1.1 Estándar 1063: “Documentación del Usuario”

El Estándar IEEE 1063 es el estándar para la Documentación del Usuario del Software.

Este estándar describe una lista de componentes para la Documentación del Usuario. Este estándar provee recomendaciones para el contenido, la organización y la presentación de dicha documentación. Esta serie de características puede llevarse a cabo del Modo Instrucciones y del Modo Referencia, ambos modos pueden intervenir en la Documentación del usuario.

El Modo Instrucciones se intenta que lo use aquella persona que necesite aprender sobre el software y el Modo Referencia intenta que el usuario solamente realice una consulta para refrescar su memoria sobre el software.

A pesar de que esta documentación es realizada por el técnico del software, debe ser revisada al final por el personal encargado de realizar las prueba del software, ya que en este documento, durante la revisión, se pueden encontrar defectos del software e incluso errores de omisión.

Este estándar define un modelo de proceso básico para identificar lo que se requiere para la Documentación del Usuario, obviamente este proceso es llevado a cabo antes del comenzar a realizar la Documentación del usuario. Los pasos de este proceso son los siguientes:

1. Identificar el software. En el se identifica la interfaz del software y las tareas que el usuario realizará con el software. Esto debe ser identificado al iniciar la planeación de la documentación, es fundamental.
2. Determinar el personal al cual será dirigido y para los cuales será realizado el documento. Se debe de identificar su nivel y el perfil de esos usuarios.
3. Determinar los documentos. En esta parte se determinan la organización de la serie de documentos que entraran dentro del Documento de Usuario.
4. Determinar el Modo del Documento de Usuario. Este puede ser, como ya vimos anteriormente, de Modo Referencia, de Modo Instrucciones o de ambos, esto es para el Documento del Usuario.

Así también este estándar define 12 componentes básicos para un Documento de Usuario del Software, y proporciona varias guías dependiendo de si es un solo documento o es un conjunto de documentos, así como también depende del tamaño de documentos.

Los componentes son los siguientes:

- Título de la página
- Restricciones del Documento
- Garantías
- Tabla de contenido (Índice)
- Lista de ilustraciones
- Introducción
- Estructura del documento (contenido, organización y presentación)
- Condiciones de error
- Apéndice
- Bibliografía
- Glosario
- Índice de referencia

La estructura de un documento de Modo Instrucción puede ser organizado de manera orientada a información o orientada a tareas específicas.

El contenido de un documento de manera orientada a tareas puede ser el siguiente:

- Alcance-metas
- Material
- Antecedentes
- Precauciones y advertencias
- Métodos
- Información relacionada

La estructura de un documento de Modo Referencia puede ser organizada de acuerdo a como el usuario va a tener acceso a alguna función del software.

Pueden ser incluido lo siguiente:

- Propósito
- Material
- Antecedentes
- Entrada de datos
- Precauciones y advertencias
- Llamadas a funciones
- Interrupción de funciones
- Finalización de funciones
- Salidas de datos

4.1.2 Estándar 803 de la IEEE: "Especificación de Requerimientos"

El Estándar IEEE 803 describe la manera recomendada para la Especificación de Requerimientos del Software.

Este estándar se basa principalmente en la respuesta a unas preguntas básicas, las cuales especifican la funcionalidad de software, sus interfaces externas, funciones básicas, atributos y el diseño implementado.

Este estándar puede estar dirigido al producto, pero no al proceso del software. Este estándar (SRS) solamente nos ayuda a especificar que debe hacer el software, más no como debemos hacerlo. Éste describe características para un buen SRS, como son:

- Correcto. Un SRS es correcto si y solo si cada requerimiento que se encuentre dentro del SRS, formará parte del software o se encontrará en éste.
- Inequívoco.- Un SRS es inequívoco si, y solo si, todos los requerimientos que se encuentren en él, tengan solamente una interpretación.
- Completo.- Un SRS es completo si, y solo si, incluye los siguientes elementos:

    ⇒ Todos los requerimientos significativos relacionados a la funcionalidad, desempeño, atributos o a las interfaces externas.

    ⇒ Definición de los resultados generados por el software, como respuesta a datos de entrada y a toda clase de situación.

⇒ Etiquetas y referencias de todas las figuras, tablas y diagramas en el SRS, así también la definición de todos los términos y unidades de medida.

- Consistente.- Si un SRS no está de acuerdo con algún documento de un nivel superior, como la especificación de requerimientos del sistema, entonces no es correcto.
- Rango de importancia y / o estabilidad.- Un SRS va a tener rango de importancia y estabilidad si cada uno de los requerimientos en éste, tiene un identificador para indicar la importancia o estabilidad particular del requerimiento.
- Verificable.- Un SRS es verificable si, y solo si, cada requerimiento que pertenecen a él es verificable. Un requerimiento es verificable cuando existe algún proceso rentable con el cual, una persona o maquina, puedan revisar que el producto de software cumpla con el requerimiento.
- Modificable.- Un SRS es modificable si, y solo si, su estructura es tal que cualquier cambio a los requerimientos puede hacerse fácilmente, completo y con consistencia, manteniendo la estructura.
- Identificable.- Un SRS es identificable si el origen de cada uno de los requerimientos es claro y esto facilita la referencia de cada requerimiento en futuros desarrollos o en la mejora de la documentación.

La documentación del usuario del software es algunas veces usada como sustituto cuando un SRS no ha sido desarrollado. Esto es muy común en un ambiente no estructurado de desarrollo de software. Es fácil ver que la documentación del usuario no puede sustituir a un SRS, sobretodo en cuanto a las características de éste último. La documentación del usuario puede ser correcta, consistente, verificable y modificable, pero casi nunca es inequívoco, completa e identificable.

El SRS juega un rol muy importante para lograr la calidad del software, este es un documento principal y fue creado para obtener el criterio por el cual la calidad será medida. Éste especifica los requerimientos del software y podría también reflejar las necesidades del cliente y el usuario, así como sus expectativas.

Desarrollar un buen SRS es un proceso importante y fundamental. Sin él se dificulta: 1. El plan y la validación del software, porque no es totalmente claro lo que debe hacer el software; 2. El manejo del proyecto del software, porque cuando hay cambios en los requerimientos se incrementa el costo-tiempo asociados, lo cual es difícil justificar.

Un SRS es requerido como mínimo para estándares de calidad del software y diferentes regulaciones. Desarrollarlo podría incrementar la calidad del producto del software, y reducir el costo-tiempo del proyecto de software.

El estándar IEEE 830 define tres categorías de personas con un directo e indirecto interés en el SRS: el cliente, el distribuidor (del software) y el usuario (el cliente y el usuario no son la misma persona).

En este caso el cliente y distribuidor pueden trabajar juntos para producir un buen escrito y entendiendo el SRS. Esto no significa que juntos deban escribir directamente el SRS, sino que el cliente dará al distribuidor por medio de documentos, entrevistas, diagramas y otros medios, la descripción de los requerimientos del software. Posteriormente, el distribuidor genera el SRS de manera formal y al terminarlo es revisado por el cliente.

El SRS es desarrollado durante la fase de análisis de requerimientos de el proyecto. Se puede decir que es la parte principal, la entrada, para comenzar con la fase de Diseño, así también es la base para la Implementación y Prueba.
Es necesario que durante las siguientes fases del proyecto, el SRS sea actualizado, ya que durante el desarrollo del software, se pueden ir encontrando defectos originales e inexactitudes. Esto es muy importante, ya que el SRS es el documento principal de todo el desarrollo, es la base de éste.

El estándar IEEE 830 también proporciona una serie de aspectos específicos para un prototipo de SRS. Este estándar se encarga de ofrecer diferentes plantillas para prototipos. También se tiene que decir que el estándar no obliga a que un SRS tenga exactamente la misma estructura de las platillas, pero eso sí, debe incluir toda la información que se presenta en éstas.

Los puntos en común y básicos para SRS son:

1. Introducción
   - 1.1 Propósito
   - 1.2 Alcance
   - 1.3 Definiciones, siglas y abreviaturas
   - 1.4 Referencias
   - 1.5 Apreciación global
2. Descripción global
   - 2.1 Perspectiva del producto
   - 2.2 Funciones del producto
   - 2.3 Características del usuario
   - 2.4 Restricciones
   - 2.5 Asunciones y dependencias
3. Requerimientos específicos

Apéndices
Índice

El punto 3 es el más importante, ya que en él se especifican los requerimientos de la organización para el proyecto de software. En esta parte podemos encontrar las características que de un SRS: correcto, inequívoco, completo, consistente, verificable e identificable.

El estándar permite la especificación de requerimientos para diferentes organizaciones, pero en éstos se intenta incluir la siguiente información:

- Interfaces externas
- Funciones
- Requerimientos de funcionamiento
- Requerimientos lógicos de bases de datos
- Diseño de restricciones
- Conformidad con los estándares o regulaciones correspondientes para el software.
- Atributos del sistema de software.

Este estándar puede facilitar hacer un SRS a un ingeniero de software sin experiencia, aunque es recomendable que en este caso, se comience trabajando con una aplicación pequeña de software.

Uno de los puntos más importantes en el estándar 830 de la IEEE es el utilizar prototipos, ya que éstos pueden ser usados para obtener o detectar los requerimientos del software, para obtener directamente algún formato de reporte o alguna pantalla y también algún requerimiento puede ser inferido al hacer experimentos con el prototipo.

Es recomendable usar prototipo en la fase de requerimientos del prototipo, la creación de un prototipo durante esta es una técnica muy efectivo para dar más confianza a un SRS, además puede disminuir el costo-tiempo.

4.1.3 Estándar 829 de la IEEE: "Documentación de prueba"

En el estándar 829 se describe la documentación prueba estándar en la colección de estándares de la IEEE.

Este estándar define los siguientes tipos de documentos de prueba:

- Prueba de plan
- Prueba de la especificación del diseño
- Prueba de la especificación del caso
- Prueba de la especificación del proceso
- Prueba del reporte de la transmisión del artículo
- Test log
- Prueba del reporte incidente
- Prueba del reporte del resumen

Dependiendo del tamaño del proyecto o de las actividades en las cuales se efectuarán pruebas, será el número de documentos prueba que se llevarán a cabo.

Las pruebas del software pueden ser realizadas en diferentes niveles, para unidades individuales del software, para un grupo de unidades, para la ejecución de la aplicación del software en una simulación de éste o para la aplicación del software integrado totalmente. El estándar 829 no define estos niveles de prueba, pero pueden ser usados los documentos de prueba del software en cada uno de estos niveles.

Es muy importante comprender los niveles de prueba en orden para poder entender el contexto del estándar 829. Para esto se tomará como base el estándar 610.12 de la IEEE (Glosario de la terminología de la ingeniería del software), para poder definir los más comunes niveles de prueba, que son los siguientes:

- Prueba de unidad: prueba individual de unidades de software o hardware o grupos de unidades relacionadas.
- Prueba de integración: prueba en la cual los componentes del software, los componentes del hardware o ambos, son probados para evaluar la interacción entre ellos.
- Prueba del sistema: prueba que se realiza con un sistema completo e integrado, para evaluar si cumple con la especificación de requerimientos.

La prueba de sistema podría ser realizada siempre, ya que es empleada para cualquier tipo de software, se puede decir que es la prueba básica para cualquier proyecto. Según el nivel de complejidad o integridad del software, se van incluyendo las otras dos pruebas. Por ejemplo, para un nivel de integridad 1, es un nivel bajo y solo aplicaremos la prueba del sistema.

Era importante hacer mención de lo anterior para cerrar más nuestro panorama de pruebas y poder conocer los niveles de prueba que se usan en el software.

Con respecto al estándar 829 de la IEEE se puede decir que define un aprovechamiento estandarizado de los documentos de prueba del software, así como también a que sean realizadas pruebas del software más sistemáticamente, permitiendo que haya pruebas más efectivas, seguimiento más efectivo, rehúso de pruebas y pruebas de procedimientos, entre otros beneficios.

El estándar 829 hace un enfoque modular para la documentación prueba del software, especificando numerosos documentos que pueden ser producidos en diferentes fases del ciclo de vida del software. Este enfoque modular puede disminuir el costo tiempo asociado con la prueba de software, puede facilitar el rehúso de casos de prueba y las pruebas de procesos en proyectos posteriores.

Este estándar está principalmente pensado en personal de prueba de software. Desarrolladores de software y personal de calidad pueden estar involucrados en la revisión de esta documentación, por lo que están en un segundo termino para utilizar este estándar, así también personal de la administración del software que está involucrada en el proceso.

Con esto podemos asegurar que el estándar 829 puede estar afectando todas las diferentes fases del ciclo de vida del software, pues en cada una de éstas, el estándar recomienda una serie de documentos de prueba para cada diferente organización.

4.1.4 Estándar 1016 de la IEEE: "Descripción del diseño"

Este estándar es recomendado para describir el diseño del software. Este estándar especifica la información necesaria que debe incluir este diseño y recomienda también la organización para la descripción del diseño de software (SDD, Software Design Description). Un SDD es una representación de un sistema, que es usado como medio para comunicar la información del diseño de software. El SDD muestra como el sistema de software será estructurado para satisfacer los requerimientos identificados. Se puede decir que es una traducción de requerimientos para poder tener una descripción de la estructura del software, los componentes del software, las interfaces y los datos necesarios para la fase de implementación. En un SDD completo, cada requerimiento debe ser identificable, ya sea uno solo o más diseño de entidades.

El diseño de entidad es un elemento de un diseño que es estructural y funcionalmente distinto a otros elementos y que es distinto en nombre y referencia. Se puede entonces dar una clasificación de diferentes diseños de entidades, en la cual preferentemente se busca dar un término genérico para unidad. Un entidad diseño puede ser una de las siguientes:

- Unidad
- Proceso
- Base de datos
- Grupo de entidades diseño de bajo nivel (sistemas, subsistemas o programas).

Debe quedar muy claro el término diseño de entidad, ya que el propósito de un SDD es describir "diseños de entidades" y éste ayudará a determinar el alcance de la unidad y la prueba de integración. El estándar 1016 proporciona las pautas para crear un SDD, lo cual trae consigo beneficios importantes, como son:

- Aseguramiento del diseño de actividades
- Facilidad el diseño de la comprobación
- Simplificación de la implementación del software
- Permisos para la unidad de software y la prueba de integración
- Mejora en el mantenimiento del software

Una de las razones importantes por la cual debemos realizar un SDD es para darnos cuenta que en el mantenimiento de un sistema de software, se asegurará que el diseño y la implementación usada para el sistema de software satisfaga los requerimientos de éste. (duda de si lo pongo o no)

Este estándar será usado principalmente por los diseñadores de software para documentar su trabajo. Otras más como los analizadores de requerimientos, los implementadores del software y el personal de pruebas trabajaran con el SDD, según lo requiera cada uno. Éste último usará el SDD para crear los diferentes documentos de prueba necesarios en los proyectos.

Se debe hacer notar que todas las fases del ciclo de vida del software van de la mano, una detrás de otra y se deben llevar a cabo en orden y todas para poder lograr un software con calidad. Ene este caso el SDD es la "salida" de la fase de diseño, y la entrada para lograr este documento es la especificación de los requerimientos, documentación creada en la fase de análisis de requerimientos.

Así también el SDD es la entrada en la fase de implementación, sin olvidar que también es una entrada para las actividades de prueba en la fase de prueba, pues los documentos de unidad y de diseño de integración dependen de las especificaciones de "la entidad de diseño" que está en un SDD.

En el estándar solamente se recomienda una solo organización para un SDD, pero es solamente una de las muchas maneras en que se puede organizar y dar formato a al información. Lo que se debe incluir en todas las diferentes formas de organizar un SDD es la descripción de los atributos o características de cada entidad. El estándar requiere los siguientes atributos para cada entidad:

- Identificación
- Tipo
- Propósito
- Función
- Dependientes (Subordinados)
- Dependencias
- Interfase
- Recursos
- Procesos
- Datos

El estándar recomienda organizar los atributos (características) del diseño de la entidad de acuerdo a la "vista del diseño", la cual se puede definir como un grupo de información sobre los atributos del diseños de la entidad, que especifica las actividades necesarias para el proyecto de software.

El estándar recomienda varias vistas de diseño, en las cuales se puede ver la descomposición, dependencia, interfase y descripción detallada.

El siguiente cuadro muestra estas vistas de diseño recomendadas por el estándar 1016:

Vistas de diseño	Alcance	Atributos de la entidad	Representaciones ejemplos
Descripción de la descomposición	Partición del sistema en diseño de entidades	Identificación, tipo, propósito, función y dependientes.	Jerarquía, diagrama de descomposición, lenguaje natural.
Descripción de la descomposición	Descripción de las relaciones entre las entidades y recursos.	Identificación, tipo, propósito, dependencias y recursos.	Estructura de los mapas, diagrama de flujo de datos y transición de diagramas.
Descripción de la interfaz	Lista de todo diseñador, programador y persona de prueba que necesita saber usar el diseño de entidades que formen el sistema.	Identificación, función e interfaces.	Los archivos de interfaz y la tabla de parámetros.
Descripción detallada	Descripción de los detalles del diseño interno de una entidad.	Identificación, proceso y dato.	Diagramas de flujo.

Es importante comentar que estas vistas pueden ser desarrolladas de manera secuencial, proporcionando cada una capa adicional de información. Así, la fase de diseño puede ser descompuesta in 4 subfases, cada una con un objetivo comprobable. Las revisiones del software pueden ser realizadas en cada vista de diseño. Esto es conveniente especialmente para proyectos muy grandes.

El estándar no exige organizar la documentación de acuerdo a la las vistas de diseño que recomienda, según la información que cada quien requiera y considere necesaria, se podrá organizar el SDD como se quiera y aún así podrá satisfacer lo requerido por el estándar.

Por último, debemos tomar en cuanta que el tipo de entidad de diseño que será descrita en un SDD, deberá ser determinada al inicio de la fase de diseño.

4.1.5 Estándar 982.1 y estándar 1061 de la IEEE: "Métricas y medidas"

Estándar 982.1-1988 de la IEEE, Standard Dictionary of Measures to Produce Reliable Software.

Estándar 1061-1988 de la IEEE, Standard for a Software Quality Metrics Methodology.

Estos estándares pueden ser usados para evaluar al mismo software o al proceso de desarrollo de software. Algunas son aplicadas en las características estáticas del software (como el análisis del código fuente) y otras en las características dinámicas en el ejecutable del software.

El uso de métricas y medidas puede incrementar la fiabilidad del producto de software y la calidad, proporcionar mejoras en las entradas de la planeación del proceso de software y ayuda en la detección de fallas en el proceso de desarrollo de software. Estos estándares proporcionan grandes beneficios para la calidad del software.

Las métricas y medidas proporcionan medios cuantitativos para evaluar el software de manera confiable y de calidad, que nos pueden dar más objetiva, tangible y neutralmente metas para el personal encargado del proyecto.

El estándar 982.1 especifica un cierto número de medidas de software. Algunas de estas medidas pueden ser usadas para evaluar productos de software y otras para evaluar el proceso de desarrollo de software.
Explícitamente el estándar se enfoca a la fiabilidad del software, el cual se entiende que es la probabilidad de que el software no cause fallas en el sistema por un tiempo determinado y bajo condiciones específicas.

Es natural el énfasis que el estándar pone en la fiabilidad del software, ya que en la relación entre los valores computarizados obtenidos con las medidas y la existencia de defectos en el sistema que aún no son descubiertos es un una probabilidad siempre inherente en la realidad.

El estándar 982.1 trata de ofrecer ciertos beneficios como por ejemplo:

- El objetivo del proceso es proporcionar medidas que puedan ser aplicables en todas las partes del ciclo de vida y también proporcionar los medios para continuar su propia evaluación y mejorar la fiabilidad.

- El objetivo del producto es incrementar la fiabilidad del software en el actual ambiente, durante la fase de operación y soporte.

Este estándar esta pensado para ser del interés del diseñador, del desarrollador, para el personal de evaluación y mantenimiento del software. También es de interés para el personal de calidad del software, para la gente de operación y soporte.

El estándar 1061 describe un modelo de proceso para establecer requerimientos para la calidad del software e identificar, implementar, analizar y validar las métricas para la calidad del software.

Este estándar presenta ciertos beneficios, tales como:

- Evaluación de logros de las metas de calidad
- Establecer requerimientos de calidad para un sistema desde el principio de éste.
- Establecer aceptar estándares
- Evaluar los niveles de calidad logrados a través de los requerimientos establecidos.
- Detectar anomalías y puntos clave de problemas en el sistema
- Predecir los niveles de calidad que podrían ser logrados en un futuro
- Monitoreo de cambios cuando el software es modificado
- Validar el conjunto de métricas.

El estándar 1061, esté pensado para el de adquisición, el desarrollador, el usuario, el de soporte, el de mantenimiento y el auditor de software.

Se puede ver que estos estándares pueden ser usado por muchas personas en el desarrollo del software, ya que hay muchos diferentes tipos de métricas y medidas que pueden ser aplicados en varios contextos.

Ambos estándares pueden presentarse durante todo el ciclo de vida del software, es decir, puede estar presente en todas las fases del proyecto de software.

Descripción

El estándar 982.1 de la IEEE define 39 diferentes medidas, incluyendo ya en este número las medidas de producto y software. El estándar clasifica las medidas de acuerdo al siguiente esquema:

Medidas de producto. La medida de producto dirige la causa y efecto de los aspectos estáticos y dinámicos de la fiabilidad del proyecto antes de la operación y la fiabilidad de la operación.
Se cuenta con 6 medidas de producto como una subclasificación, para obtener las dimensiones de la fiabilidad:

- Errores, faltas, fracasos. Contador de defectos por causa huma, de bichos de programa y malos funcionamientos observados en el sistema.
- Medida-tiempo-fracasos, rango de fracaso. Medidas derivadas de la ocurrencia y el tiempo.
- Fiabilidad del crecimiento y proyección.
- Faltas en el producto.
- Integridad y consistencia.
- Complejidad. Valor de los valores que se complican ene le sistema.

Medidas de procesos. Las tres medidas de proceso con las que se cuenta como subclasificación son las siguientes:

- Gerencia de control. El valor de guiar el desarrollo y mantenimiento del proceso.
- Cobertura. La valoración de la presencia de todas las actividades necesarias para el desarrollo y el mantenimiento del producto de software.
- Riesgo, beneficio, evolución del costo. La valoración de los tratados del proceso de costo, horario y aplicación.

El estándar 982.1 proporciona un modelo de “Proceso de medida de la fiabilidad”, que consiste de 8 fases. Aunque este modelo es útil, resumido muy bien vía un flujo-mapa del proceso, es muy similar al modelo de proceso definido en el estándar 1061 de la IEEE.

Este último estándar (**1061 de la IEEE**) , define una metodología de métricas para la calidad de software. Este es un modelo proceso de 5 fases o pasos, para especificar los requerimientos de calidad y las métricas de calidad que son usadas para la evaluación. Nos son especificadas métricas, pero pueden ser usadas las medidas del estándar 982.1 o pueden definirse métricas de calidad para un producto en específico.

Las 5 fases de la metodología son descritas como actividades, y las voy a citar como sigue:

1. Establecer los requerimientos de la calidad del software
2. Identificar las métricas de calidad del software
3. Implementar las métricas de calidad de software
4. Analizar los resultados de las métricas del software
5. Validación de las métricas de calidad del software

4.2 Estándares de proceso

Los estándares de proceso son más difíciles de implementar que los estándares del producto del trabajo (descritos en el siguiente capítulo). Esto es porque los estándares de proceso son menos tangibles, éstos suponen que ya han sido implementados antes algunos estándares de producto de trabajo. En cambio, los estándares de producto de trabajo describen documentos específicos o aspectos del mismo software y así no requieren una estructura conceptual.

4.2.1 Estándar 1058 de la IEEE: "Administración de proyecto"

Estándar 1058-1998 de la IEEE, Estándar para la planeación de la administración del proyecto.

Un Plan para la Administración del proyecto des software (SPMP) es definido por el estándar como sigue:

> *"Este plan define la parte técnica y administrativa del proceso, necesaria para desarrollar los productos del trabajo de software que satisfagan los requerimientos del producto."*

El tamaño o tipo de producto de software no limita la aplicación de este estándar. Proyectos pequeños podrán requerimientos formalidad en la planeación que en los proyectos grandes, pero todos los componentes del estándar deberán ser incluidos en todo proyecto del software.

El estándar 1058 puede disminuir los costos y tiempo del proyecto, e incrementar la calidad del producto. Permite hacer más efectiva la administración de los proyecto durante el ciclo de vida del software. Así también, el estándar proporciona lo necesario para crear un correcto plan.

Este estándar está pensado para ser utilizado por los administradores del proyecto de software y personal que prepara y actualiza los planes de proyecto.

Este estándar puede ser aplicado a cualquier o a todas las fases de un ciclo de vida del software. El SPMP podrá se restringido a cualquier fase del ciclo de vida.

Descripción

El estándar define dos tipos de complacencia, la formato y la de contenido. En el primero el formato exacto y el contenido del estándar son seguidos en el plan de proyecto. En el de contenido los contenidos de este estándar son reestructurados en el plan de proyecto.

Para resumir los problemas de complacencia, toda la información contenida en el SPMP del estándar deberá ser incluida, aunque el formato puede ser reemplazado y se pueden adicionar secciones de documentos adicionales.

El formato del plan de administración del proyecto de software, establecido por el estándar 1058 de la IEE es como se muestra a continuación:

Titulo de la página
identificador de página
Cambios en la historia
Prefacio
Contenido de tablas
Lista de figuras
Lista de tablas

1. Apreciación global
   1.1 Resumen del proyecto
   1.2 Evolución del plan
2. Referencias
3. Definiciones
4. Organización del proyecto
   4.1 Interfaces externas
   4.2 Estructura interna
   4.3 Roles y responsabilidades
5. planes para la administración del proceso
   5.1 Plan salida
   5.2 Plan de trabajo
   5.3 Plan de control
   5.4 Plan de administración de riesgos
   5.5 Cierre del plan
6. Planes técnicos del proceso
   6.1 Modelo de proceso
   6.2 Métodos, herramientas y técnicas
   6.3 Plan de infraestructura
   6.4 Plan de aceptación del producto
7. Planes del proceso de soporte
   7.1 Plan de administración de la configuración
   7.2 Plan de verificación y validación
   7.3 Plan de documentación
   7.4 Plan del aseguramiento de calidad
   7.5 Revisiones y auditorias
   7.6 Plan de solución de problemas
   7.7 Plan de administración del subcontratante
   7.8 Plan de mejora del proceso
8. Planes adicionales

Anexos
Índice

Así se puede ver que el estándar propone una serie de pasos o fases necesarias para tener un correcto plan de administración para el producto de software.
A grandes rasgos se explicarán los puntos principales de este plan.

La cláusula 4 describe las interfaces externas de las entidades dentro de la organización, la estructura interna del proyecto y las responsabilidades de los elementos internos de la organización con mayor actividades de trabajo.

La cláusula 5 es usada para describir cosas como el personal, establecer el tiempo y el presupuesto.

En la cláusula 6 se definirá el ciclo de vida del software, el desarrollo de métodos, se describirá como establecer el ambiente de desarrollo del producto y especifica el proceso y el criterio para el cliente para aceptar el proyecto liberado.

La cláusula 7 cuenta con los planes para los proceso de soporte. En esta cláusula en especifico pueden ser aplicados varios estándares, como por ejemplo el estándar 828 (Estándar de la IEEE para el plan de la administración de la configuración del software), el 1012 (estándar de la IEEE para la verificación y validación del software) y el 730 (estándar de la IEEE para el plan de aseguramiento de calidad del software).

La cláusula 8 se separa de la 7 porque podría haber otros planes que podrían jugar otro rol importante (que no entrara dentro de ninguno de los anteriores) para el ciclo de vida del software o podría no abarcar la duración del proyecto de software.

Un SPMP es un documento completo y es el principal mecanismo de control para el proyecto de software.

4.2.2 Estándar 1028 de la IEEE: “Revisiones del software”

Estándar 1028-1997 de la IEEE, Estándar para la revisión del software”

El término “Revisiones del software”, abarca los conceptos de revisiones, inspecciones y auditorias.

El propósito de este estándar es definir revisiones aplicables a la adquisición, la entrega, el desarrollo, la operación y el mantenimiento del software. En pocas palabras, describe como llevar a cabo una revisión

El estándar hace mucho énfasis en que estas revisiones son *revisiones sistemáticas*, esto implica que debe intervenir la participación de un equipo, debe haber resultados documentados de la revisión y procedimientos documentados para dirigir la revisión.

El estándar define 5 tipos de revisiones de software:

- Revisión de la Administración
- Revisión técnica
- Inspección
- Seguimiento
- Auditoria

Para cada uno de estos tipos de revisión el estándar especifica que se debe generar un reporte de cada uno: Reporte de revisión de la Administración, Reporte de la revisión técnica, Reporte de la inspección, Reporte del seguimiento y Reporte de la auditoria.

El estándar se refiere, en cuanto a las revisiones, a los productos de software. Entendemos como producto de software un conjunto completo de programas, procedimientos , documentación y datos asociados, o también, puede ser uno o más términos independientes de este conjunto anterior.

Es importante definir el producto de software, ya que el estándar usa este término en un sentido muy amplio, pues en esto se basa para generar sus revisiones. Algunos ejemplo de este términos incluidos en el estándar son: Reportes de anomalías, Reportes de auditoria, Procedimientos, Planes de contingencia.

Las revisiones del software, es una de las actividades más importantes de validación y verificación (V&V), como lo es también las pruebas de software. Gracias a la evaluación de los “productos de trabajo” en el ciclo de vida del software, algunas anomalías, requerimientos omisos o incorrectos, pueden ser identificados y corregidos tempranamente. Esto reduce los costos y tiempo de un proyecto. En muchos casos, la revisiones pueden encontrar defectos que habrían sido difícil detectar por medio de las pruebas o comprobación. Sin embargo con esto se puede ver que aplicando ambos procesos es seguro poder obtener un producto final con mucho más calidad.

Este estándar es usado en muchas circunstancias. Es usado en lo administrativo y lo técnico, para el desarrollo, la comprobación, y con el personal encargado de la calidad. Así también el cliente , el distribuidor y el usuario final, puede intervenir en las revisiones del software, pues al final de cunetas son las personas que verán el resultado y dirán si cumple o no con lo requerido y lo establecido al comienzo del proyecto.

Por lo anterior también podemos afirmar que las revisiones del software son incluidas en todas las fases del proyecto.

Descripción

El estándar 1028 especifica la siguiente información para cada tipo de revisión, se puede decir que cada una de las revisiones debe contener lo siguiente:

- Introducción.- el propósito general de la revisión, una descripción global y ejemplos de productos de software convenientes para este tipo de revisión.
- Responsabilidades.- los roles y responsabilidades de los participantes en la revisión.
- Entradas.- las entradas necesarias para la revisión.
- Criterio de entrada.- Condiciones requeridas para comenzar las revisiones, incluyendo la autorización y el inicio del evento.
- Procedimientos.- los paso para la revisión, incluyendo la administración, la planeación, la descripción global, la preparación y la evaluación.
- Criterios de salida.- condiciones requeridas para completar la revisión.
- Salidas.- es el conjunto de resultados entregados a partir de la revisión.

Revisiones de la administración

El personal que participa en esta revisión tienen roles específicos, como son: Personal de toma decisión, el líder de revisión, registrador (supervisor), personal administrativo, personal técnico y el representante del usuario o cliente.
En esta revisión el responsable de evaluar que las metas de la revisión sean llevadas acabo adecuadamente es el personal administrativo, quien desde un principio especifica estos objetivos.

Un reporte de revisión de la administración será hecho por el líder de revisión y deberá contener la siguiente información de acuerdo a lo que establece el estándar:

a) El proyecto deber ser revisado
b) La revisión de los miembros del equipo
c) Revisión de objetivos
d) El producto de software revisado
e) Las entradas de la revisión
f) El estatus de los temas de acción, propiedad y tiempo asignado o tiempo de realización.
g) Una lista de anomalías identificadas por el equipo de revisión que deberán ser utilizadas para poder cumplir las metas del proyecto.

Las decisiones tomadas en una revisión de administración pueden ser tomadas directamente durante la revisión. Las decisiones tomadas en la revisión de la administración están enfocadas en los problemas de administración usualmente involucrados en el proceso del ciclo de vida del software.

Revisiones técnicas

Los roles del personal en las revisiones técnicas es el mismo que en las revisiones de administración, pero existen algunas tareas diferentes para la parte técnica y administrativa. El staff administrativo puede participar en la revisión, con el propósito de identificar problemas que requieran soluciones administrativas. El staff técnico participa en la revisión y evaluación del producto de software.

Un las objetivos de una revisión técnica es generar una lista de acciones, enfatizando los riesgos.

Un reporte de revisión técnica podrá ser creado por el líder de revisión y deberá contener la siguiente información de acuerdo a lo que establece el estándar:

a) El proyecto debe ser revisado
b) Revisión de los miembros del equipo
c) Revisión del producto de software
d) Especificar las entradas para la revisión
e) Revisión de los objetivos y si fueron cumplidos
f) Una lista de las anomalías del producto resultas o sin resolver
g) Una lista de anomalías del hardware o acciones específicas sin resolver del sistema
h) Una lista de problemas administrativos
i) El estatus de los temas de acción, propiedad y tiempo asignado o tiempo de realización
j) Recomendaciones hechas por el equipo de revisión para problemas y anomalías
k) Especificar si el producto de software cumple con regulaciones, estándares, guías, planes y procedimientos sin corromper.

Las proceso de decisión para una revisión técnica es más restringida que para la revisión administrativa, ya que la información y recomendaciones son por la administración (dirección), quienes toman las decisiones en base a estas entradas. Es decir, la revisión técnica depende de la parte administrativa.

Inspecciones

El rol de los participantes en una inspección es muy diferente a las anteriores. En una inspección hay líder de inspección, supervisor (encargado de estar presente en la revisión, hace recomendaciones hechas durante la revisión y es el encargado de proporcionar esta información al líder) , lector (encargado de leer todo el contenido de la revisión), autor (prepara al producto de software para la inspección y proporciona información necesaria de éste durante la inspección) y el inspector (identifica y describe anomalías en el producto de software).

La inspección es más estructurada que la revisión administrativa o la revisión técnica, ya que se deben cubrir grandes cantidades de información técnica dentro de cierto tiempo establecido, esto para evitar problemas relacionados a la dinámica de equipo de revisión.

Un reporte de inspección deberá ser emitido por el líder de inspección, y deberá contar con la siguiente información de acuerdo a lo que dice el estándar:

a) El proyecto deberá ser inspeccionado
b) Inspección de los miembros del equipo
c) Duración de la inspección
d) El producto de software inspeccionado
e) El tamaño de los materiales inspeccionados
f) Especificar las entradas de la inspección
g) Revisión de los objetivos y si fueron cumplidos
h) Lista de las anomalías, localización de cada una de éstas, descripción, y clasificación
i) El resumen de las anomalías en la inspección, enlistando el número de anomalías de acuerdo a su clasificación.
j) La distribución del producto de software
k) Un estimado del esfuerzo y de la fecha de realización

La disposición del producto de software estará basada en el número y gravedad de las anomalías identificadas y la estimación del esfuerzo de trabajo.

Las anomalías podrían clasificarse según el impacto causado en el producto de software.

El proceso de decisión de la inspección es diferente a la de la revisión técnica que está autorizada a tomar decisiones en caso de que lo requiera el trabajo en el producto de software. Con esto también tiene la capacidad de poder delegar responsabilidades a el personal técnico.

Seguimiento

Los participantes tiene diferentes roles en este tipo de revisión , pero son parecidas a las anteriores: líder de seguimiento, supervisor, autor y miembros del equipo.

Esta es una técnica de análisis en la cual el diseñador o el programador del equipo de desarrollo y otras partes interesadas en el producto de software, hacen preguntas y comentarios sobre los posibles errores, la violación de los estándares de desarrollo y otros problemas.

El reporte de seguimiento, al igual que los anteriores, será realizado por el líder de seguimiento y deberá contener la siguiente información de acuerdo al estándar:

a) El seguimiento de los miembros del equipo
b) La evaluación del producto de software
c) Los objetivos logrados durante el seguimiento
d) Una lista de recomendaciones hechas para cada anomalía
e) Una lista de acciones, fechas y personas responsables
f) Algunas recomendaciones hechas por el equipo de seguimiento en cuanto a las deficiencias y las anomalías no resueltas
g) Algunas propuestas hechas por el equipo de seguimiento para que siga a ésta

El proceso de decisión del seguimiento es más restringido que la inspección, ya que el seguimiento está más orientado a proporcionad información y menos orientado a trabajar en las anomalías.

Auditorias

Para este tipo de revisión existen tres diferentes roles para el personal: líder de auditoria, supervisor y auditoria.

Una meta principal de evaluación de las auditorias es el hacer y documentar observaciones. Esto es una característica particular de los auditores porque es requerida la independencia y objetividad de los auditores.

Un reporte de auditoria podría ser creado por el líder de auditoria, y deberá contener la siguiente información, según el estándar:

a) El propósito y alcance de la auditoria
b) Organización auditada, incluyendo localización, enlace de staff y dirección.
c) Identificación del producto de software auditado
d) La aplicación de regulaciones, estándares, guías, planes y procedimiento usados para la evaluación.
e) Criterio de evaluación
f) Resumen de la organización de auditoria
g) Resumen de las actividades para examinar
h) Resumen del plan de actividades para desarrollar no realizado
i) Lista de observaciones clasificadas de mayor a menor
j) Un resumen e interpretación de los resultados de la auditoria incluyendo los temas importantes inconformes
k) El tipo de actividades siguientes a la auditoria

Información adicional, como recomendaciones de la organización de auditoria, puede también ser incluida en el Reporte de Auditoria.

4.2.3 Estándar 730 de la IEEE: "Aseguramiento de calidad"

Estándar 730-1998 de la IEEE, Planeación para el aseguramiento de calidad del software.

Este estándar especifica los requerimiento mínimos de información y el formato que deben contener un Plan de Aseguramiento de Calidad del Software (SQAP: siglas en inglés). A pesar de que esto es representado por una estándar de producto de trabajo, también sabemos que el producto de trabajo está funcionando como un plan y esto implícitamente especifica requerimientos del proceso.

Esto se hace más claro, pues se sabe que el estándar especifica documentos que han sido generados durante el proyecto, las mínimas revisiones y auditorias dirigidas durante el proyecto también y así sucesivamente. Es importante hacer notar esto, porque este estándar pareciera a simple vista ser un estándar de producto de software, y aunque si lo es, es más de proceso, ya que para cumplir con los requisitos del estándar, se deben tomare en cuenta varias elementos obtenidos durante el proceso.

Es un hecho que el QA (Aseguramiento de Calidad) es más amplio, más complejo y es un tema de más alto nivel que la validación y verificación (V&V). De hecho, las actividades de verificación y validación están estipuladas por el SQAP.

El estándar 730 aplica en el desarrollo y el mantenimiento un software crítico. Un software crítico es un software si fracasara, impactaría en la seguridad, el ambiente financiero y en la seguridad social.

Los mínimos requerimientos especificados por el estándar 730 son:

1. Documentación
2. Estándares, prácticas, convenciones y métricas
3. Revisiones y auditorias

Este estándar podría enfocarse en estos puntos para un software crítico. Así también , cuando el software puede ser particionado, alguno de estos punto puede tener un objetivo específico en esa parte critica del software.

La mínima documentación requerida (para un software critico) especificada por el estándar 730 es la siguiente:

1. Especificación de requerimientos del software (Estándar 830 de la IEEE)
2. Descripción del diseño de software (Estándar 1016 de la IEE)
3. Plan de verificación y validación del software (Estándar 1012 de la IEEE)
4. Reporte de verificación y validación del software (Estándar 1012 de la IEEE)
5. Documentación del usuario (Estándar 1063 de la IEEE)

6. Plan de administración de la configuración del software (Estándar 828 de la (IEEE)

También especifica el grupo mínimo de estándares, prácticas, convenciones y métricas (para un software crítico), que es el siguiente:

1. Estándares de documentación
2. Estándares de la estructura lógica
3. Estándares de codificación
4. Estándares de comentarios
5. Estándares de prueba y práctica (Estándar 829 de la IEEE)
6. Métricas para la calidad del software (Estándar 1061 de la IEEE)

Y por último, especifica los requerimiento mínimo de revisión y auditoria (para un software crítico), y son

1. Revisión de los requerimientos de un software
2. Una revisión preliminar del diseño
3. Revisión del diseño crítico
4. Revisión del plan de verificación y validación del software
5. Auditoria funcional
6. Auditoria física
7. Auditorias en el proceso
8. Revisiones administrativas
9. Revisión del plan de la administración de la configuración
10. Revisión post-mortem

Las revisiones y auditorias pueden ser dirigidas en paralelo con el estándar 1028 de la IEEE (Estándar de revisiones del software).

Se puede decir, que el principal beneficio de esté estándar (como de la mayoría de los estándares) es mejorar la calidad del producto de software, que se tiene esperado desde su implementación.
La implementación adecuada del proceso requerido por este estándar, se verá en el reducido costo y tiempo, además mejorar la operatividad del proyecto de software.

En cuanto al personal que interviene en este estándar se encuentran tres grupos. Uno es el usuario, el desarrollador y el público (quizás aquel que es afectado por la operación de los usuarios del producto).
El estándar también podría ser del interés ciertos grupos dentro de una organización, como por ejemplo de la administración, de la ingeniería de software, del aseguramiento de calidad, del grupo de prueba de software, del de verificación y validación, de la administración de configuración y de los escritores técnicos.

El estándar es aplicable a todas las fases del proyecto. Todas las actividades realizadas durante el aseguramiento de calidad deben ser hechas durante el ciclo de vida del software. El plan de aseguramiento de calidad deberá crearse a la salida del proyecto de software, antes del proceso de desarrollo.

Existe un formato para el plan de aseguramiento de calidad del software que recomienda el estándar 730, este formato contiene los siguientes puntos:

1. Propósito
2. Documentos referencia
3. Administración
4. Documentación
5. Estándares, prácticas, convenciones y métricas
6. Revisiones y auditorias
7. Pruebas
8. Reporte del problema y la acción correctiva
9. Herramientas, técnicas y métodos
10. Control del código
11. Medio de control
12. Proveedor de control
13. Colección de archivos, mantenimiento y periodos de retensión
14. Entrenamiento
15. Riesgos administrativos

Es recomendado usar el formato del documento SQAP exactamente como lo especifica el estándar 730, para evitar que la información y la tabla requerida sea de otra manera.

4.2.4 Estándar 1012 de la IEEE: "Verificación y validación"

Estándar 1012-1998 de la IEEE, estándar de verificación y validación del software.

Es importante, antes de comenzar con la descripción del estándar, entender le significado de ambos términos de este estándar. Se tomará la definición del estándar 610.12 de la IEEE (Estándar de la terminología de ingeniería en software), pues se considera que es más clara:

Validación.- es el proceso de valuación de un sistema o de un componente durante o al final del proceso de software para determinar si satisface los requerimientos especificados.

Verificación.- es el proceso de valuación de un sistema o de un componente para determinar si los productos de una fase de desarrollo dada satisface las condiciones determinadas al inicio de la fase.

Los proceso de validación proporciona el apoyo para comprobar que el software satisfaga los requerimientos establecidos y resuelva los problemas de riesgo. El estándar describe los procesos para este propósito y especifica el formato y contenido del *Plan de Verificación y Validación del Software* (SVVP) y del *Reporte de Verificación y Validación del Software* (SVVR).

Estos procesos son:

1. Procesos administrativo
2. Procesos de adquisición
3. Procesos de abastecimiento
4. Procesos de desarrollo
5. Procesos de operación
6. Procesos de mantenimiento

Se tiene que cada uno de estos procesos es descrito en términos de actividades, pero el proceso de desarrollo está compuesto de múltiples actividades, que son:

1. Actividad concepto
2. Actividad de requerimientos
3. Actividad de diseño
4. Actividad de implementación
5. Actividad de prueba
6. Actividad de instalación y chequeo

Para cada una de las actividades en cada proceso, el estándar define ciertas tareas de V&V (validación y Verificación) como una función del nivel de integridad del software. Este término es definido por el estándar con referencia al ISO/IEC 15026, que dice que el nivel de integridad es el rango de valores de la propiedad de un elemento necesaria para mantener los riesgos del sistema dentro de limites aceptables. Es importante tomar en cuenta este rango, pues nos va a ser un indicador del bueno o mal funcionamiento de algún elemento del software.
Los niveles de integridad deberán ser asignados a requerimientos individuales o a módulos de software. El estándar define los niveles de integridad alto, mayor, moderado y bajo.

Un riesgo basado en el nivel de integridad del software tiene también ciertas consecuencias, descritas en el Anexo B del estándar y que son: catastrófica, critica, marginal y despreciable.

Las consecuencias de los posibles errores son combinadas con el estimado de probabilidad de ocurrencia para calcular el nivel de integridad del software.

Así entonces, el nivel de integridad determina las tareas requeridas de verificación y validación. El estándar describe los procesos, actividades y tareas que son requeridas según el nivel de integridad del software, y éstas son:

1. Propósito
2. Documentos de referencia
3. Definiciones
4. Apreciación global de V&V
5. Procesos de V&V
6. Requerimientos reportados en la V&V
7. Requerimientos administrativos en la V&V

8. Documentación de requerimientos de la V&V

Si es requerido incluir los reportes en el punto 7, estos pueden ser 4 tipos diferentes, especificados en el estándar, pero no se describe su formato:

1. Reporte de tareas de V&V
2. Reportes del resumen de actividades de la V&V
3. Reporte de las anomalías en la V&V
4. Reporte final de la V&V

Al igual que los demás estándares citados en esta tesis, el estándar 1012 tiene importantes beneficios, uno de ellos y el más importante es incrementar la calidad del software, lo cual suena repetitivo (en la mayoría de los estándares se resalta este beneficio) pero se considera que es el principal objetivo de cualquier estándar de software, y es el principal objetivo de esta tesis: mostrar una serie de guías para lograr tener un software con la mayor calidad posible.
Además este beneficio, el estándar tiene otros beneficios más particulares como el facilitar la detección y corrección de errores en el software y refuerce la visión del riesgo en el producto y en el proceso.

El estándar tiene participantes específicos para su uso, están los proveedores, el cliente, los diseñadores, el personal de mantenimiento, los practicantes de V&V, los operadores, y los directivos tanto del los proveedores como de los clientes.

4.2.4 Estándar 828 de la IEEE: "Administración de la configuración"

El estándar 828 de la IEEE es para el Plan de la Administración de la Configuración del Software.

Este estándar define a un Administrador de la Configuración del Software (SCM, siglas e inglés). El propósito del SCM es manejar los cambios de la configuración de artículos (CI) y el estándar define este último término como un agregado de hardware, de software o de ambos que es designado para la administración de configuración y es tratado como una entidad en el proceso de administración de la configuración.

El CI puede ser como algo "entregable" producido durante el proceso de desarrollo de software. Documentos como el SRS y el SDD, el código fuente, o código ejecutable pueden ser identificados como CI's.

El proceso de SCM consiste de las siguiente actividades:

1. Identificación de la configuración
2. Control de la configuración
3. Estado de la contabilidad
4. Configuración de auditorias y revisiones

El estándar 828 especifica las actividades a realizar por el SMC durante el ciclo de vida del software, que son las siguientes:

- Identificación de la configuración
- Control de la configuración
- Configuración de la contabilidad
- Configuración de las revisiones y auditorias
- Control de la interfase
- Control del contratante y vendedor

También la actividad de control requiere cuatro actividades adicionales, que son:

- Cambios requeridos
- Evaluación de cambios
- Aprobación o desaprobación de cambios
- Implementación de cambios

El estándar 828 también especifica las secciones que debe contener un SCMP, y son las siguientes:

- Introducción
- Administración de un SCM
- Actividades del SCM
- Horarios del SCM
- Recursos de SCM
- Plan de mantenimiento para el SCM

La administración de la configuración del software puede ser de mucha utilidad para controlar los costos y los tiempos de una organización.

El personal de administración del software y el personal técnico para aplicar el estándar, ellos son los grupos principales de personal para crear y actualizar la configuración de los artículos durante el ciclo de vida del software.

El SCMP podría ser desarrollado en paralelo con el SPMP (Plan de la Administración del Proyecto de Software). La administración de la configuración del software puede ser realizada en el desarrollo del proceso del software, durante las fases de operación y mantenimiento.

CONCLUSIONES

En la actualidad se puede ver que la única forma de competir en los mercados internacionales es con calidad en el desarrollo de software, solo así se logrará el éxito en la producción de *software.* Esto sólo es posible con la implantación de un Sistema para el Aseguramiento de la Calidad del *Software* directamente relacionado con la política establecida para su elaboración y que esté en correspondencia con la definición internacional ISO de calidad.

Así surgen modelos como el CMM, como respuesta a la problemática actual del desarrollo de software. El CMM proporciona una estructura conceptual y metodológica para mejorar la gerencia y el desarrollo de S/W y, por ende, la calidad de los productos. Se sabe que el CMM es un camino, pero no es el único.

Los estándares son como guías establecidas para ayudarnos a cumplir con lo que estipulan los modelos de proceso y las normas de calidad. Los estándares engloban los “conocimientos” que son patrimonio de una organización, proporcionan un marco para implementar procedimientos de aseguramiento de la calidad y proporcionan continuidad entre el trabajo de distintas personas

Necesitamos que las empresas pierdan el temor de establecer procesos por medio de los diferentes estándares, que se cuente con el apoyo de los directivos, que se proporcione entrenamiento y educación a todos los niveles de la organización, y sobre todo, ir aprendiendo de los errores.

Poco a poco se verán resultados y se logrará la calidad en el desarrollo de software que se desea.

BIBLIOGRAFÍA

- ***Ingeniería del Software: Teoría y práctica***
  Shari Lawrence Pfleeger
  Edit. Pentice Hall
  1a. Edición, 2002
  Pag. 1-13 y 51-67

- ***Software Engineering***
  Ian Sommerville
  Edit. Addison Wesley
  6a. Edición, E.U. 2001
  Pag. 5-12, 30, 63-67

- ***Implementing the IEEE Software Engineering Standars***
  Michael Schmidt
  Edit. SAMS
  E.U., 2002

- ***Ingeniería del Software: Un enfoque práctico***
  Roger S. Pressman
  Edit. Mc. Graw Hill
  5a. Edición, 2002
  Capítulos 1, 2, 3, 4.

- ***Software Engineering***
  Gregory W. Jones
  Edit. John Wiley & Sans
  E.U., 1990
  Pag. 11-25

- ***Fundamentals of Software Engineering***
  Carlo Ghezzi, Mehdi Jazayeri y Dino Mandrioli
  Edit. Prentice may
  E.U., 1991
  Pag. 1-19

REFERENCIAS

CMMI

- www.esi.es/en/Training/docum/cmmiDirectivos.pdf
- www.infocalidad.net/secciones/030103.pdf

ESTÁNDARES DE SOFTWARE

- qwww.asq.org
- qwww.software.org
- http://www.ati.es/gt/calidad-software/presentacion.htm
- http://www.pcm.gob.pe/portal_ongei/publica/metodologias/Lib5086/indice.htm
- http://www.ieee.org

ISO 9000

- http://www.calidad.com.ar/iso9000.html
- http://www.calidad.com.ar/iso9000.html
- http://www.calidad.com.ar/calid503.html
- http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/caliso9mil.htm#ten
- http://www.calidad.com.ar/calid052.html

CALIDAD DE SOFTWARE

- http://www.bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm

INGENIERÍA DEL SOFTWARE

- http://www.acm.org/crossroads/espanol/xrds7-4/intro74.html
- http://dis.um.es/~jnicolas/09BK_FIS.html
- http://mail.udlap.mx/~tesis/lis/garcia_r_ci/capitulo2.pdf