Universidad Politécnica de Chiapas

Ingeniería en desarrollo de software

Luis Fernando Hernández Morales

Ingeniería de software

Cuatrimestre 6

09 de mayo de 2018

Suchiapa, Chiapas

ISO/IEC 12207

Estándar para los procesos de ciclo de vida del software de la organización ISO. Para conseguirlo, el estándar se basa en dos principios fundamentales: modularidad y responsabilidad. Con el modularidad se pretende conseguir procesos con un mínimo acoplamiento y una máxima cohesión. En cuanto a la responsabilidad, se busca establecer un responsable para cada proceso, facilitando la aplicación del estándar en proyectos en los que pueden existir distintas personas u organizaciones involucradas, no importando el uso que se le dé a este.

Los procesos se clasifican en tres tipos: procesos principales, procesos de soporte y procesos de la organización. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto ejecutado. Los procesos principales se instancian de acuerdo con la situación particular.

- Procesos principales
    - Adquisición
    - Suministro
    - Desarrollo
    - Operación
    - Mantenimiento
    - Destrucción
- Procesos de soporte
    - Documentación
    - Gestión de la configuración
    - Aseguramiento de la calidad
    - Verificación
    - Validación
    - Revisión conjunta
    - Auditoría
    - Resolución de problemas
- Procesos de la organización
    - Gestión
    - Infraestructura
    - Mejora
    - Recursos humanos

ISO/IEC TR 15504

También conocido como Software Process Improvement Capability Determination, abreviado SPICE, en español, «Determinación de la Capacidad de Mejora del Proceso de Software» es un modelo para la mejora, evaluación de los procesos de desarrollo, mantenimiento de sistemas de información y productos de software. El proyecto SPICE tenía tres objetivos principales:

Desarrollar un borrador de trabajo para un estándar de evaluación de procesos de software.

Llevar a cabo los ensayos de la industria de la norma emergente.

Promover la transferencia de tecnología de la evaluación de procesos de software a la industria del software a nivel mundial.

Establece un marco y los requisitos para cualquier fase de evaluación de procesos y proporciona requisitos para los modelos de evaluación de estos. Proporciona también requisitos para cualquier modelo de evaluación de organizaciones. Proporciona guías para la definición de las competencias de un evaluador de procesos. Actualmente tiene 10 partes: de la 1 a la 7 completas y de la 8 a la 10 en fase de desarrollo. Comprende: evaluación de procesos, mejora de procesos, determinación de capacidad. Proporciona, en su parte 5, un Modelo de evaluación de procesos para las fases de ciclo de vida del software definidos en el estándar ISO/IEC 12207 que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de los sistemas de software. Proporciona, en su parte 6, un Modelo de evaluación de procesos para las etapas de ciclo de vida del sistema, definidos en el estándar ISO/IEC 15288 que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de sistemas. Proporcionará, en su parte 8, un Modelo de evaluación de procesos para los procesos de servicios TIC que serán definidos en el estándar ISO/IEC 20000-4 que definirá los procesos contenidos en la norma ISO/IEC 20000-1.

Tiene una arquitectura basada en dos dimensiones: de proceso y de capacidad de proceso. Define que todo modelo de evaluación de procesos debe definir: - la dimensión de procesos: el modelo de procesos de referencia (dimensión de las abscisas) - la dimensión de la capacidad: niveles de capacidad y atributos de los procesos. Los niveles de capacidad para todo modelo de evaluación de procesos pueden tener desde el 0 y al menos hasta el nivel 1 de los siguientes niveles de capacidad estándar:

- Nivel 0: Incompleto
- Nivel 1: Realizado
- Nivel 2: Gestionado
- Nivel 3: Establecido
- Nivel 4: Predecible
- Nivel 5: En optimización

Para cada nivel existen unos atributos de procesos estándar que ayudan a evaluar los niveles de capacidad.

SEI (Instituto de Ingeniería de Software)

Es un instituto federal estadounidense de investigación y desarrollo, fundado por Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa de los Estados Unidos y administrado por la Universidad Carnegie Mellon.

El Modelo de Madurez de Capacidades o CMM (Capability Maturity Model), es un modelo de evaluación de los procesos de una organización. Fue desarrollado inicialmente para los procesos relativos al desarrollo e implementación de software por la Universidad Carnegie-Mellon para el Software Engineering Institute (SEI).

Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area). Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser:

- Definidas en un procedimiento documentado
- Provistas (la organización) de los medios y formación necesarios
- Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas)
- Medidas
- Verificadas

A su vez estas Áreas de Proceso se agrupan en cinco "niveles de madurez", de modo que una organización que tenga institucionalizadas todas las prácticas incluidas en un nivel y sus inferiores, se considera que ha alcanzado ese nivel de madurez.

1 - Inicial. Las organizaciones en este nivel no disponen de un ambiente estable para el desarrollo y mantenimiento de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven minados por falta de planificación. El éxito de los proyectos se basa la mayoría de las veces en el esfuerzo personal, aunque a menudo se producen fracasos y casi siempre retrasos y sobrecostes. El resultado de los proyectos es impredecible.

2 - Repetible. En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente.

3 - Definido. Además de una buena gestión de proyectos, a este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detalladas y un nivel

más avanzado de métricas en los procesos. Se implementan técnicas de revisión por pares (peer reviews).

4 - Gestionado. Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.

5 - Optimizado. La organización completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

Así es como el modelo CMM establece una medida del progreso, conforme al avance en niveles de madurez. Cada nivel a su vez cuenta con un número de áreas de proceso que deben lograrse. El alcanzar estas áreas o estadios se detecta mediante la satisfacción o insatisfacción de varias metas claras y cuantificables.

Capability Maturity Model Integration

Integración de modelos de madurez de capacidades o Capability Maturity Model Integration (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Las mejores prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay tres áreas de interés cubiertas por los modelos de CMMI: Desarrollo, Adquisición y Servicios. La versión actual de CMMI es la versión 1.3 la cual fue liberada el 1 de noviembre de 2010.

Nivel 1: No Confiable- Ambiente impredecible donde las organizaciones no tienen actividades de control y no están diseñadas.

Nivel 2: Informal- Las actividades de control existen, pero no se ponen en práctica. Los controles dependen básicamente de las personas. No hay un entrenamiento formal ni comunicación de las actividades de control.

Nivel 3: estandarizado- Las actividades de control existen y están diseñadas, han sido documentadas y comunicadas a los empleados, las desviaciones de las actividades de control probablemente no se detecten.

Nivel 4: Monitoreado- Se utilizan herramientas en una forma limitada para soportar las actividades de control

Nivel 5: Optimizado- Es una estructura integrada de control interno con un monitoreo en tiempo real por la gerencia, así como mejoras continuas-auto control, se

encuentran cambios más rápidos al momento de detectar errores en los manejos de las actividades o en las personas.

IEEE/Std. 830

Es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales. Además de los casos de uso, la ers también contiene requisitos no funcionales (complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación, como, por ejemplo, restricciones en el diseño o estándares de calidad.

Está dirigida tanto al cliente como al equipo de desarrollo. El lenguaje utilizado para su redacción debe ser informal, de forma que sea fácilmente comprensible para todas las partes involucradas en el desarrollo.

Las características de una buena ERS son definidas por el estándar IEEE 830-1998. Una buena ERS debe ser:

Completa. Todos los requerimientos deben estar reflejados en ella y todas las referencias deben estar definidas.

Consistente. Debe ser coherente con los propios requerimientos y también con otros documentos de especificación.

Inequívoca. La redacción debe ser clara de modo que no se pueda mal interpretar.

Correcta. El software debe cumplir con los requisitos de la especificación.

Trazable. Se refiere a la posibilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.

Priorizable. Los requisitos deben poder organizarse jerárquicamente según su relevancia para el negocio y clasificándolos en esenciales, condicionales y opcionales.

Modificable. Aunque todo requerimiento es modificable, se refiere a que debe ser fácilmente modificable.

Verificable. Debe existir un método finito sin costo para poder probarlo.

Tipos de requisitos:

Requisitos de Usuarios: Necesidades que los usuarios expresan verbalmente

Requisitos del Sistema: Son los componentes que el sistema debe tener para realizar determinadas tareas

Requisitos Funcionales: Servicios que el sistema debe proporcionar al finalizar el sistema.

IEEE/Std. 1362

El ConOps es un documento el cual se desarrolla con el fin de aclarar qué es, lo que realmente desea el cliente y que le ofrecemos en el software que desarrollaremos, por supuesto, este documento debe estar firmado por ambas partes como un punto de acuerdo.

Algunas de las características fundamentales de este documento son:

- Que es lo que realmente se necesita (prioridades) y cómo hacerlo.

- Descripción de la situación actual y sus antecedentes, por ejemplo: qué ventajas tiene el sistema y en que está fallando o que desean mejorar.

- Que restricciones hay que manejar e incorporar.

- El porqué de los cambios.

- Que mejoras se quieren hacer.

- Definir el sistema propuesto definiendo claramente en que se mejora el software o sistema propuesto y como lo hace.

Todo esto debe estar en palabras no muy técnicas para una mejor comprensión del usuario ya que si colocamos lenguaje técnico, puesto que no todos los usuarios tienen conocimiento sobre ellas y lo único que lograremos es confundir al usuario.

IEEE/Std. 1063

El estándar IEEE Std 1063-2001 brinda ese marco de referencia para establecer qué partes deben conformar cualquier documento que deba ser utilizado por un usuario del sistema o programa en cuestión.

Este estándar solo se aplica a la documentación de usuario, para la documentación de carácter técnico se utilizan otras recomendaciones de las cuales hablaremos en su momento. A grandes rasgos la recomendación del IEEE establece las siguientes partes para un documento que usarán los usuarios. Estas partes son:

1.- Identificación de los datos (paquete, título)

2.- Tabla de contenidos, en documentos con más de 8 páginas

3.- Lista de ilustraciones (optativo)

4.- Introducción

5.- Información para el uso de la documentación

6.- Conceptos de las operaciones

7.- Procedimientos

8.- Información sobre los comandos de software

9.- Mensajes de error y resolución de problemas

10.- Glosario

11.- Referencias

12.- Características de navegación

13.- Índice o Índex

14.- Herramientas de búsqueda (en documentos electrónicos).

IEEE/Std. 1012

Estándar de verificación de software.

Consiste en la verificación y validación de un software, es un procedimiento que está basado en normas de calidad en algunos modelos de vida de un software.

Se aplica al software adquirido, desarrollado, mantenido o modificado, el cual está ligado al ciclo de vida en cada una de las etapas o procesos que lleva el software.

Sus objetivos son proporcionar una evaluación objetiva de los productos y procesos del ciclo de vida del software, evidencia si los requisitos de sistema son apropiados, completos, precisos, coherentes y comprobables.

Facilita la detención de anomalías del software, mejora la visión de la gestión de riesgos, mejora la gestión de la visión de riesgos, apoya las mejoras de procesos para generar un modelo de análisis de un sistema integrado.

VERIFICACIÓN	VALIDACIÓN
Cumplir con las normas prácticas y convenciones.	Satisfacer los requisitos del software asignado al final de cada ciclo de vida.
Completar con éxito la actividad.	Satisfacer el uso del usuario.
Cumplir con los requisitos.	Solucionar con exactitud el problema.

IEEE/Std. 1219

Es el único estándar que íntegramente se ocupa del proceso de mantenimiento del software. Describe un proceso iterativo para la gestión y ejecución de las actividades del proceso. Aunque sólo menciona las fases de desarrollo y de producción de un producto de software, éstas cubren todo su ciclo de vida, cualquiera que sea su tamaño o complejidad.

Fases de la Norma

Esta norma define cambios en un producto de software a través de un proceso de mantenimiento dividido en fases, el proceso es iterativo y en cascada, con una gran semejanza al ciclo de vida del desarrollo clásico, como se menciona a continuación:

- Identificación del problema.
- Análisis.
- Diseño.
- Implementación.
- Pruebas del sistema.
- Pruebas de aceptación.
- Puesta en producción o liberación de versión.

Según el IEEE, el mantenimiento de software es la modificación del producto después de su entrega, con el objetivo de corregir defectos, para mejorar el rendimiento u otras propiedades deseables.

Dentro de cada una de estas fases, el estándar define una serie de procedimientos que se han de llevar a cabo y con los que se identifica la documentación, las personas y productos de software que intervienen.

Esta norma plantea un proceso de mantenimiento con gran nivel de detalle y documentación a llevar para su desarrollo, haciéndolo muy útil y necesario sobre todo en los lugares que se realiza mantenimiento del software, aquí es fundamental la traza que marca el estado y evolución de cada una de las fases, pero pudiera resultar excesivo para pequeñas organizaciones que deseen aplicar dicho estándar en el mantenimiento de sus sistemas internos.

Tipos de mantenimiento:

- ✓ Mantenimiento adaptativo
  - Tiene por objetivo la modificación de un programa debido a cambios en el entorno, ya sea hardware o software en el que se ejecuta.
- ✓ Mantenimiento correctivo
  - Tiene por objetivo localizar y eliminar los posibles defectos del programa.
- ✓ Mantenimiento de emergencia
  - Es un mantenimiento correctivo realizado sin planificación previa.

- ✓ Mantenimiento preventivo
  - Consiste en la modificación del producto sin alterar las especificaciones del mismo, para mejorar las propiedades de mantenimiento del producto y facilitar así las futuras tareas de mantenimiento.
- ✓ Mantenimiento perfectivo
  - Es la modificación del producto después de su puesta en producción y para mejorar el rendimiento o la mantenibilidad, que es la facilidad de mantenimiento que tiene un software, es decir, la facilidad para ser modificado.