# EL SOFTWARE Y LA INGENIERÍA DE SOFTWARE

LA NATURALEZA DEL SOFTWARE

* Requiere tiempo para ser terminado
* Es por sobre todo complejo
* No podemos detectar todos los errores
* Dificultades para medir el avance mientras se desarrolla

Definición de software

El software es elemento de un sistema lógico. Por lo tanto posee características

1. El software se desarrolla o modifica con intelecto: Los costos del software se concentran en la ingeniería

2. El software no se “desgasta”. ¡pero sí se deteriora!

Durante su vida el software sufre cambios, y se introducen errores que ocasionan que la curva tenga aumentos súbitos. Poco a poco, el nivel de tasa de fallas comienza a aumentar: el soft se esta deteriorando.

3. Aunque la industria se mueve hacia la construcción basada en componentes, la mayor parte del software se construye para un uso individualizado.

Un componente de software debe diseñarse e implementarse de modo que pueda volverse a usar en muchos programas diferentes.

**Dominios de aplicación del software**

7 categorias:

Software de sistemas: conjunto de programas escritos para dar servicio a otros programas.

Software de aplicación: programas aislados que resuelven una necesidad específica de negocios.

Software de ingeniería y ciencias: se ha caracterizado por algoritmos “devoradores de números”.

Software incrustado: reside dentro de un producto o sistema y se usa para implementar y controlar características y funciones para el usuario final y para el sistema en sí.

Software de línea de productos: es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes. Control de inventarios, Mercados masivos.

Aplicaciones web: Webapps

Software de inteligencia artificial: hace uso de algoritmos no numéricos para resolver problemas complejos

Computación en un mundo abierto: desarrollar software de sistemas y aplicación que permita a dispositivos móviles, computadoras personales y sistemas empresariales comunicarse a través de redes enormes.

Construccion de redes: WWW, que es un proveedor de contenido

Fuente abierta: Distribucion de código fuente para aplicación de sistemas.

Software heredado

Programas antiguos, por lo general de mala calidad y con diseños que no son susceptibles de extenderse, código confuso, documentación mala o inexistente, casos y resultados de pruebas que nunca se archivaron

Este tipo de software tiende a evolucionar por las sig. Razones:

- debe adaptarse a nuevos ambientes de la tecnología

- debe ser operable con otros sistemas o bases de datos modernos

- debe ser viable dentro de un ambiente de redes

LA NATURALEZA ÚNICA DE LAS WEBAPPS

La gran mayoría de webapps presenta los siguientes atributos:

- Uso intensivo de redes. Una webapp reside en una red y debe atender las necesidades de una comunidad diversa de clientes.

- Concurrencia: gran nro de usuarios a la vez

- carga impredecible: El nro de usuarios de la webapp cambia repentinamente.,

- rendimiento

- Disponibilidad

- Orientado a objetos: presentar al usuario final contenido en forma de texto, gráficas, audio y video. Además, las webapps se utilizan en forma común para acceder a información que existe en bases de datos.

- Contenido sensible: La calidad y la estética son importantes en una webapp

- Evolucion continua

- Inmediatez

- Seguridad: deben implementarse medidas estrictas de seguridad a través de la infraestructura de apoyo de una webapp y dentro de la aplicación misma.

- Estetica

INGENIERÍA DE SOFTWARE

- El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas. Debe hacerse un esfuerzo concertado para entender el problema antes de desarrollar una aplicación de software.

- Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa. El diseño se ha vuelto una actividad crucial.

- Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas

- el software debe tener facilidad para recibir mantenimiento.

La ingeniería de software es: 1) La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.

EL PROCESO DEL SOFTWARE

Accion: Conjunto de tareas.

Tarea: objetivo pequeño pero bien definido.

Una estructura de proceso general para la ingeniería de software consta de cinco actividades:

- Comunicación: Comunicarse y colaborar con el cliente. Entender los objetivos y reunir los requerimientos.

- Planeacion: describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

- Modelado: crear modelos a fin de entender mejor los requerimientos del software y el diseño

- Construcción. Esta actividad combina la generación de código y las pruebas

- Despliegue: entrega , evaluación y retroalimentación.

El proceso de software es iterativo e incremental.

el proceso de ingeniería de software son complementadas por cierto número de actividades sombrilla. Estas actividades son:

- Seguimiento y control del proyecto de software: permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto

- **Administración del riesgo:** evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto.

- Aseguramiento de la calidad del software: define y ejecuta las actividades requeridas para garantizar la calidad del software.

- Revisiones técnicas: Evalua los productos a fin de descubrir y eliminar errores

- Medición: define y reúne mediciones del proceso

- Administración de la configuración del software: administra los efectos del cambio a lo largo del proceso del software.

- Administración de la reutilización: define criterios para volver a usar el producto

- Preparación y producción del producto del trabajo

LA PRÁCTICA DE LA INGENIERÍA DE SOFTWARE

La esencia de la práctica:

**1.** *Entender el problema* (comunicación y análisis).

**2.** *Planear la solución* (modelado y diseño del software).

**3.** *Ejecutar el plan* (generación del código).

**4.** *Examinar la exactitud del resultado* (probar y asegurar la calidad).

Principios generales

- Primer principio: La razón de que exista todo: El soft existe por una razón. La base de este principio es plantearse si lo que hacemos agrega valor real al sistema.

- Segundo principio: MSE (Mantenlo sencillo, estúpido…). Todo diseño debe ser tan simple como sea posible, pero no más.

- Tercer principio: Mantener la visión. mantener la visión y que obligue a su cumplimiento garantiza un proyecto de software muy exitoso.

- Cuarto principio: Otros consumirán lo que usted produce. siempre establezca especificaciones, diseñe e implemente con la seguridad de que alguien más tendrá que entender lo que usted haga.

- Quinto principio: Ábrase al futuro. los sistemas de software con verdadera “fortaleza industrial” deben durar mucho más tiempo. Nunca diseñe sobre algo iniciado.

- Sexto principio: Planee por anticipado la reutilización. La reutilización ahorra tiempo y esfuerzo.

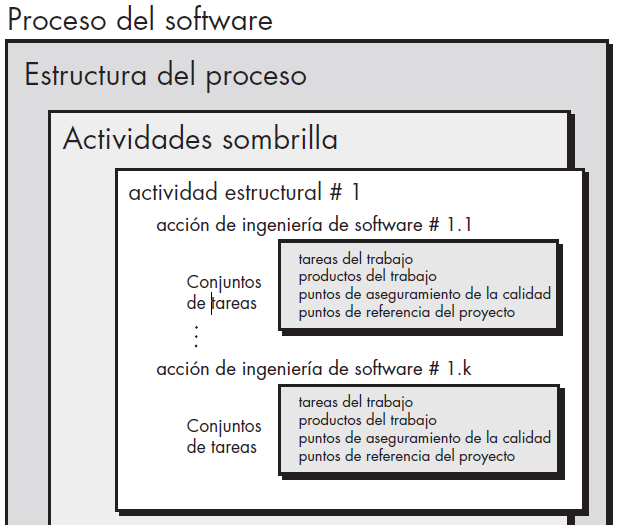
- Séptimo principio: ¡Piense! Pensar en todo con claridad antes de emprender la acción casi siempre produce mejores resultados.

# MODELOS DEL PROCESO

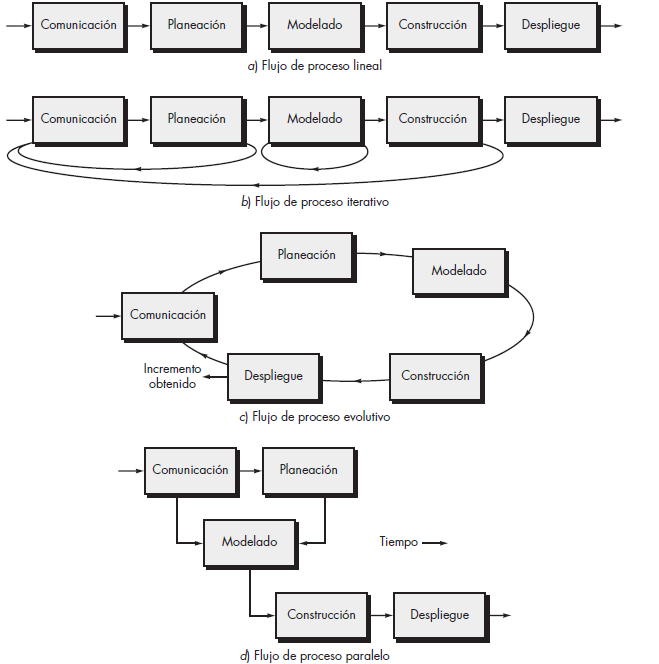
se define proceso del software como una estructura para las actividades, acciones y tareas que se requieren a fin de construir software de alta calidad.

se representa el proceso del software de manera esquemática. En dicha figura, cada actividad estructural está formada por un conjunto de acciones de ingeniería de software y cada una de éstas se encuentra definida por un conjunto de tareas que identifica las tareas del trabajo que deben realizarse.

para la ingeniería de software define cinco actividades estructurales: comunicación, planeación, modelado, construcción y despliegue. Además, a lo largo de todo el proceso se aplica un conjunto de actividades sombrilla.



El flujo del proceso se describe como la manera en que están organizadas las actividades estructurales y las acciones y tareas



Definición de actividad estructural

Surge una pregunta clave: ¿qué acciones son apropiadas para una actividad estructural, dados la naturaleza del problema por resolver, las características de las personas que hacen el trabajo y los participantes.

Cada una de estas acciones de la ingeniería del software tendrá muchas tareas de trabajo y un número grande de diferentes productos finales.

Identificación de un conjunto de tareas

Debe escogerse el conjunto de tareas que se adapte mejor a las necesidades del proyecto y a las características del equipo. Esto implica que una acción de la ingeniería de software puede adaptarse a las necesidades

Patrones del proceso

Describe un problema relacionado con el proceso durante el trabajo de ingenieroa, identifica el ambiente y sugiere soluciones. Un patrón de proceso da un formato , un método para describir soluciones.

El formato debe tener estos datos:

- Nombre del patron

- Fuerzas: El ambiente en el que se encuentra el patrón y los aspectos que afectan su solución.

- tipo:

1\_ Patron de etapa: asociado con una actividad estructural

2\_ Patron de tarea: define un problema asociado con una acción o tarea

3\_ Patron de Fase: define la secuencia de las actividades estructurales. Ej.: Modelo espiral/prototip

- Contexto inicial. Describe las condiciones en las que se aplica el patrón.

- Problema

- Solucion

- Contexto resultante. Describe las condiciones que resultarán una vez que se haya implementado

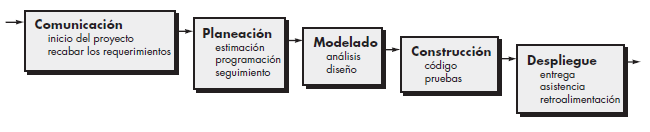
con éxito el patrón

- Patrones relacionados

MODELOS DE PROCESO PRESCRIPTIVO

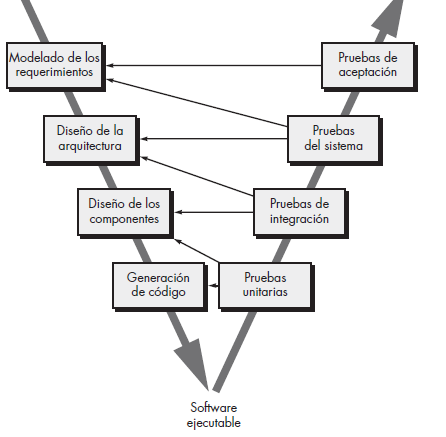
Buscan generar estructura y orden. Prescriben un conjunto de elementos del proceso: actividades estructurales, acciones de ingeniería de software, tareas, productos del trabajo, aseguramiento de la calidad y mecanismos de control del cambio para cada proyecto. Cada modelo del proceso también prescribe un flujo del proceso.

Modelo de la cascada



Sugiere un enfoque sistemático y secuencial.

Una variante de la representación del modelo de la cascada se denomina modelo en V. donde se aprecia la relación entre las acciones para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana. hacia abajo desde el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema y de su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, y en esencia ejecuta una serie de pruebas (acciones para asegurar la calidad).



Entre los problemas de este medolo tenemos:

1\_ Los proyectos no suelen adoptar un flujo secuencial

2\_ es difícil para el cliente enunciar en forma explícita todos los requerimientos

3\_ No se dispondrá de una versión funcional hasta que el proyecto esté muy avanzado

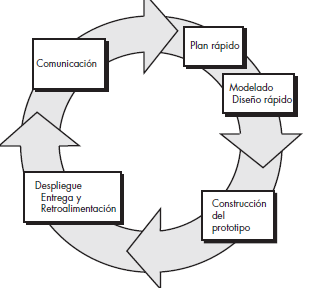
Modelos de proceso incremental

El modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce “incrementos” de software. en cada incremento se entrega un producto que ya opera.

Modelos de proceso evolutivo

Proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo. Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas

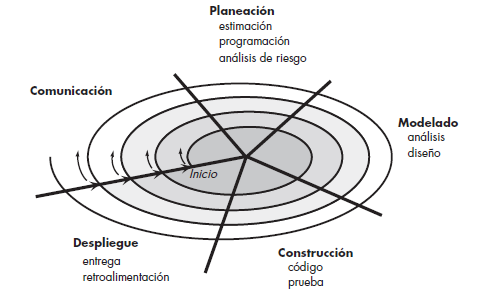
Proceso evolutivo: PROTOTIPOS



El prototipo sirve como “el primer sistema”. Aunque algunos se construyen para ser desechables.

Proceso evolutivo: ESPIRAL

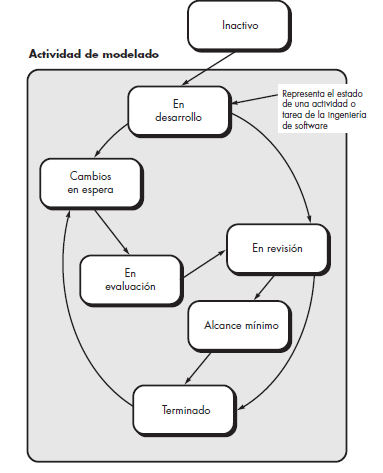
Es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas.



Modelos concurrentes

permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso. invocar una o más de las siguientes acciones de software: hacer prototipos, análisis y diseño.

Todas las actividades de ingeniería de software (Modelado, comunicación, construcción , etc) existen de manera concurrente, pero se hallan en diferentes estados.



MODELOS DE PROCESO ESPECIALIZADO

Desarrollo basado en componentes

incorpora muchas de las características del modelo espiral. el modelo de desarrollo basado en componentes construye aplicaciones a partir de fragmentos de software prefabricados.

incorpora las etapas siguientes:

1. Se investigan y evalúan, para el tipo de aplicación de que se trate, productos disponibles

basados en componentes.

2. Se consideran los aspectos de integración de los componentes.

3. Se diseña una arquitectura del software para que reciba los componentes.

4. Se integran los componentes en la arquitectura.

5. Se efectúan pruebas exhaustivas para asegurar la funcionalidad apropiada.

El modelo de métodos formales

agrupa actividades que llevan a la especificación matemática. Por medio del empleo de notación matematica. Es utilizado para construir Software de primera calidad de seguridad como control de aeronaves o equipos médicos.

Desarrollo de software orientado a aspectos

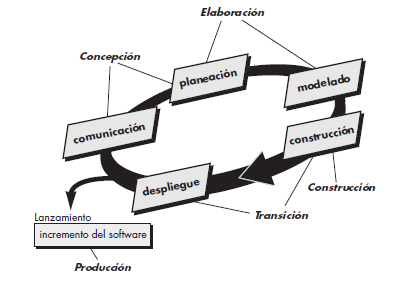
proporciona un proceso y enfoque metodológico para definir, especificar, diseñar y construir aspectos: “mecanismos más allá de subrutinas y herencia para localizar la expresión de una preocupación global”.

EL PROCESO UNIFICADO

un proceso del software “impulsado por el caso de uso, centrado en la arquitectura, iterativo e incremental”.

UML, lenguaje de modelado unificado, que contiene una notación robusta para el modelado y desarrollo de los sistemas orientados a objetos. El UML brinda la tecnología necesaria para apoyar la práctica de la ingeniería de software.

Fases del proceso unificado



La fase de concepción del PU agrupa actividades tanto de comunicación con el cliente como de planeación. Al colaborar con los participantes, se identifican los requerimientos del negocio, se propone una arquitectura aproximada para el sistema y se desarrolla un plan para la naturaleza iterativa e incremental.

La fase de elaboración incluye las actividades de planeacion y modelado del modelo general del proceso. La elaboración mejora y amplía los casos de uso.

La fase de construcción del PU es idéntica a la actividad de construcción definida para el proceso general del software. Con el uso del modelo de arquitectura como entrada, la fase de construcción desarrolla o adquiere los componentes del software.

La fase de transición del PU incluye las últimas etapas de la actividad general de construcción y la primera parte de la actividad de despliegue general (entrega y retroalimentación). Se da el software a los usuarios finales para las pruebas beta, quienes reportan tanto los defectos como los cambios necesarios. Además, el equipo de software genera la información de apoyo necesaria (por ejemplo, manuales de usuario, guías de solución de problemas, procedimientos de instalación, etc.)

La fase de producción del PU coincide con la actividad de despliegue del proceso general. Durante esta fase, se vigila el uso que se da al software, se brinda apoyo para el ambiente de operación (infraestructura) y se reportan defectos y solicitudes de cambio.

MODELOS DEL PROCESO PERSONAL Y DEL EQUIPO

Proceso personal del software (PPS)

El PPS enfatiza la necesidad de detectar pronto los errores; de igual importancia es entender los tipos de ellos que es probable cometer. Esto se logra a través de una actividad de evaluación rigurosa ejecutada para todos los productos del trabajo que se generen. El PPS representa un enfoque disciplinado basado en la medición para la ingeniería de software.

Proceso del equipo de software (PES)

es construir un equipo “autodirigido” para el proyecto, que se organice para producir software de alta calidad.

# CONCEPTOS DE CALIDAD

La calidad es un concepto complejo y de facetas múltiples:

la calidad del diseño incluye el grado en el que el diseño cumple las funciones y características especificadas en el modelo de requerimientos. La calidad de la conformidad se centra en el grado en el que la implementación se apega al diseño.

satisfacción del usuario = producto que funciona + buena calidad + entrega dentro del presupuesto y plazo

CALIDAD DEL SOFTWARE

1. Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.

2. Un producto útil entrega contenido, funciones y características que el usuario final desea

3. Al agregar valor para el productor y para el usuario de un producto, el software de alta

calidad proporciona beneficios a la organización

Dimensiones de la calidad de Garvin

- Calidad del desempeño: El software entrega las características especificadas como parte del modelo de requerimientos

- Calidad de las características. El software tiene características que sorprenden y agradan

- Confiabilidad. El software proporciona todas las características y capacidades sin fallar

- Conformidad. El software concuerda con los estándares locales y externos

- Durabilidad. El software puede recibir mantenimiento (cambiar) o corregirse

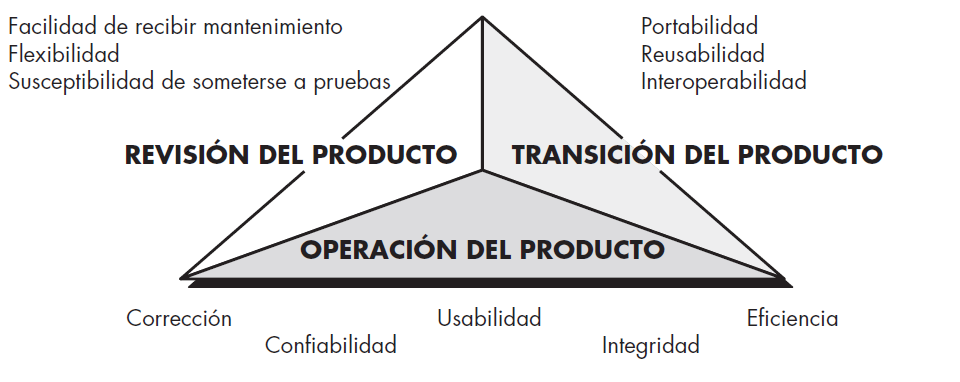
- Servicio: Correcciones o cambios en un tiempo breve

- Estetica

- Percepción. En ciertas situaciones, existen prejuicios que influirán en la percepción de la

calidad por parte del usuario. La reputación de la empresa es uno de ellos

Deben hacerse mediciones: debe compararse el software con algún dato para llegar a un indicador de la calidad.



Factores de la calidad de McCall

se centran en tres aspectos importantes del producto de software: sus características operativas, su capacidad de ser modificado y su adaptabilidad a nuevos ambientes.

- Corrección. Grado en el que un programa satisface sus especificaciones

- Confiabilidad. Grado en el que se espera que un programa cumpla con su función y con la precisión requerida

- Eficiencia. Cantidad de recursos de cómputo y de código requeridos

- Integridad. Grado en el que es posible controlar el acceso de personas no autorizadas al software o a los datos.

- Usabilidad. Esfuerzo que se requiere para aprender, operar, preparar las entradas e interpretar las salidas de un programa.

- Facilidad de recibir mantenimiento

- Flexibilidad. Esfuerzo necesario para modificar un programa

- Susceptibilidad de someterse a pruebas. Esfuerzo que se requiere para probar un programa

- Portabilidad. Esfuerzo necesario para transferir el programa de un ambiente de sistema de hardware

- Reusabilidad

- Interoperabilidad. Esfuerzo requerido para acoplar un sistema con otro.

Factores de la calidad ISO 9126: identificar los atributos clave del software

- Funcionalidad. Grado en el que el software satisface las necesidades planteadas según las establecen los atributos siguientes: adaptabilidad, exactitud, interoperabilidad, cumplimiento y seguridad.

- Confiabilidad. Cantidad de tiempo que el software se encuentra disponible para su uso, según lo indican los siguientes atributos: madurez, tolerancia a fallas y recuperación.

- Usabilidad. Grado en el que el software es fácil de usar, según lo indican los siguientes subatributos: entendible, aprendible y operable.

- Eficiencia. Grado en el que el software emplea óptimamente los recursos del sistema, según lo indican los subatributos siguientes: comportamiento del tiempo y de los recursos.

- Facilidad de recibir mantenimiento

- Portabilidad

EL DILEMA DE LA CALIDAD DEL SOFTWARE

Software “suficientemente bueno”: Se mejora a través de versiones, pero cumple con las principales funcionalidades requeridas por los usuarios.

El costo de la calidad

El costo de la calidad incluye todos los costos en los que se incurre al buscar la calidad o al realizar actividades relacionadas con ella y los costos posteriores de la falta de calidad. El costo de la calidad puede

dividirse en los costos que están asociados con la prevención, la evaluación y la falla.

- Los costos de prevención incluyen: lo siguiente: 1) el costo de las actividades de administración requeridas para planear y coordinar todas las actividades de control y aseguramiento de la calidad, 2) el costo de las actividades técnicas agregadas para desarrollar modelos completos de los requerimientos y del diseño, 3) los costos de planear las pruebas y 4) el costo de toda la capacitación asociada con estas actividades.

- Los costos de evaluación incluyen • El costo de efectuar revisiones técnicas

• El costo de recabar datos y unidades de medida para la evaluación

• El costo de hacer las pruebas y depurar

Los costos de falla son aquellos que se eliminarían si no hubiera errores antes o después de enviar el producto a los consumidores. Los costos de falla se subdividen en internos y externos. Se incurre en costos internos de falla cuando se detecta un error en un producto antes del envío.

Los costos Internos son: Reparaciones para corregir errores, efectos colaterales de las reparaciones por errores, colección de unidades de medidas que evalúen los modos de falla.

Los costos externos de falla se asocian con defectos encontrados después de que el producto se envió a los consumidores: solución de quejas, devolución y sustitución del producto, ayuda en línea y trabajo asociado con la garantía, La mala reputación y perdida de nuevos negocios.

Calidad y seguridad

el software que no tiene alta calidad es fácil de penetrar por parte de intrusos y, en consecuencia, el software de mala calidad aumenta indirectamente el riesgo de la seguridad, con todos los costos y problemas que eso conlleva.

El efecto de las acciones de la administración

- Decisiones de estimación. En muchos casos, hay una presión enorme del tiempo para entrar al mercado que fuerza al equipo a aceptar fechas de entrega irreales.

- Decisiones de programación. Como se programa la secuencia de tareas según el proyecto. Esto aveces hace que la tarea C quede dependiendo de la finalización de la tarea A.

- Decisiones orientadas al riesgo. se necesita saber lo que puede salir mal y establecer un plan de contingencia para ese caso.

LOGRAR LA CALIDAD DEL SOFTWARE

cuatro actividades principales que ayudan al equipo de software a lograr una alta calidad en éste: métodos de la ingeniería de software, técnicas de administración de proyectos, acciones de control de calidad y aseguramiento de la calidad del software:

Métodos de la ingeniería de software

Se debe entender el problema que se quiere resolver. También debe ser capaz de crear un diseño que esté de acuerdo con el problema

Técnicas de administración de proyectos

1) un gerente de proyecto usa estimaciones para verificar que las fechas pueden cumplirse, 2) se comprenden las dependencias de las actividades programadas y el equipo resiste la tentación de usar atajos, 3) la planeación del riesgo.

Control de calidad

Asegurar que todo producto del trabajo cumpla sus metas de calidad. descubrir y corregir errores antes de que comiencen las pruebas. Se aplica una serie de etapas de prueba para detectar los errores en procesamiento lógico, manipulación de datos y comunicación con la interfaz.

Aseguramiento de la calidad

La administración racional de proyectos y las acciones de control de calidad. Además, el aseguramiento de la calidad consiste en un conjunto de funciones de auditoría y reportes para evaluar la eficacia y completitud de las acciones de control de calidad.

# ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

El aseguramiento de la calidad del software (con frecuencia llamado administración de la calidad) es una actividad sombrilla que se aplica en todo el proceso del software.

La gente que realiza el ACS debe ver al software desde el punto de vista del cliente. ¿El software cumple adecuadamente los factores de calidad mencionados en el capítulo 14? ¿El desarrollo del software se condujo de acuerdo con estándares preestablecidos? ¿Las disciplinas técnicas han cumplido con sus roles como parte de la actividad de ACS? El grupo de ACS trata de responder éstas y otras preguntas para garantizar que se mantenga la calidad del software.

ELEMENTOS DE ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

- Estandares: El trabajo del ACS es asegurar que los estándares que se hayan adoptado se sigan

- Revisiones y auditorias: son un tipo de revisión efectuada por personal de ACS con objeto de garantizar que se sigan los lineamientos de calidad en el trabajo de la ingeniería de software.

- Pruebas: tiene un objetivo principal: detectar errores. El trabajo del ACS es garantizar que las pruebas se planeen en forma apropiada y que se realicen con eficiencia

- Colección y análisis de los errores: El ACS reúne y analiza errores y datos acerca de los defectos para entender mejor cómo se cometen los errores y como eliminarlos

- Administracion del cambio

- Educacion: es clave para proponer y patrocinar programas educativos.

- Administracion de los proveedores: El trabajo de la organización de ACS es garantizar que se obtenga software de alta calidad a partir de las sugerencias de prácticas específicas de calidad que el proveedor debe seguir

- Administracion de la seguridad: instituir políticas para proteger los datos en todos los niveles, establecer cortafuegos de protección para las webapps y asegurar que el software no va a ser vulnerado

- Seguridad: El ACS es responsable de evaluar el efecto de las fallas del software y de dar los pasos que se requieren para disminuir el riesgo.

- Administracion del riesgo: el ACS garantiza que las actividades de administración de riesgos se efectúen en forma apropiada

TAREAS, METAS Y MÉTRICAS DEL ACS

Tareas del ACS

Planeación, supervisión, registro, análisis y elaboración de reportes para el aseguramiento de la calidad. Estas acciones son realizadas (o facilitadas) por un grupo independiente de ACS que hace lo siguiente:

- Prepara el plan de ACS para un proyecto.

- Participa en el desarrollo de la descripción del software del proyecto.

- Revisa las actividades de la ingeniería de software a fin de verificar el cumplimiento mediante el proceso definido para el software.

- Audita los productos del trabajo de software designados para verificar que se cumpla con aquellos definidos como parte del proceso de software.

- Asegura que las desviaciones en el trabajo de software y sus productos se documenten y manejen de acuerdo con un procedimiento documentado.

- Registra toda falta de cumplimiento y la reporta a la alta dirección.

Metas, atributos y métricas

- Calidad de los requerimientos. El ACS debe garantizar que el equipo de software ha revisado en forma

apropiada el modelo de requerimientos

- Calidad del diseño. El ACS busca atributos del diseño que sean indicadores de la calidad.

- Calidad del código. El ACS debe identificar aquellos atributos que permitan hacer un análisis razonable de la calidad del código.

- Eficacia del control de calidad. El ACS analiza la asignación de recursos para las revisiones y pruebas a fin de evaluar si se asignan en la forma más eficaz.

ASEGURAMIENTO ESTADÍSTICO DE LA CALIDAD DEL SOFTWARE

El aseguramiento estadístico de la calidad implica los pasos siguientes:

1. Se recaba y clasifica la información acerca de errores y defectos del software.

2. Se hace un intento por rastrear cada error y defecto hasta sus primeras causas

3. Con el uso del Principio de Pareto (80 por ciento de los defectos se debe a 20 por ciento de todas las causas posibles), se identifica 20 por ciento de las causas de errores y defectos (las pocas vitales).

4. Una vez identificadas las pocas causas vitales, se corrigen los problemas

Ejemplo general

Aunque se descubren cientos de problemas diferentes, todos pueden rastrearse hasta una (o más) de las causas siguientes:

• Especificaciones erróneas o incompletas (EEI)

• Mala interpretación de la comunicación con el cliente (MCC)

• Desviación intencional de las especificaciones (DIE)

• Violación de los estándares de programación (VEP)

• Error en la representación de los datos (ERD)

• Interfaz componente inconsistente (ICI)

• Error en el diseño lógico (EDL)

• Pruebas incompletas o erróneas (PIE)

• Documentación inexacta o incompleta (DII)

• Error en la traducción del lenguaje de programación del diseño (LPD)

• Interfaz humano/computadora ambigua o inconsistente (IHC)

• Varios (V)

Para aplicar el ACS estadístico se elabora una tabla con el % de cada uno de los errores mencionados.

Seis Sigma para la ingeniería de software

es una metodología rigurosa y disciplinada que usa datos y análisis estadísticos para medir y mejorar el desempeño operativo de una compañía, identificando y eliminando defectos en procesos de manufactura y servicios. También seis desviaciones estándar, es La metodología que se define en tres etapas fundamentales:

• Definir los requerimientos del cliente y los que se le entregan

• Medir el proceso existente

• Analizar las métricas de los defectos y determinar las pocas causas vitales.

Si se trata de un proceso de software existente que se requiere mejorar, Seis Sigma sugiere dos etapas adicionales:

• Mejorar el proceso, eliminando las causas originales de los defectos.

• Controlar el proceso para asegurar que el trabajo futuro no vuelva a introducir las causas de los defectos.

Si una organización va a desarrollar un proceso de software (en vez de mejorar uno existente), a las etapas fundamentales se agregan las siguientes:

• Diseñar el proceso para 1) evitar las causas originales de los defectos y 2) cumplir los requerimientos del cliente.

• Verificar que el modelo del proceso en realidad evite los defectos y cumpla los requerimientos del cliente.

CONFIABILIDAD DEL SOFTWARE

La confiabilidad del software, a diferencia de muchos otros factores de la calidad, se midey estima directamente mediante el uso de datos históricos del desarrollo. La confiabilidad delsoftware se define en términos estadísticos como “la probabilidad que tiene un programa decómputo de operar sin fallas en un ambiente específico por un tiempo específico”

Mediciones de la confiabilidad y disponibilidad

una medida sencilla de su confiabilidad es el tiempo medio entre fallas (TMEF):

TMEF = TMPF + TMPR

donde las siglas TMPF y TMPR significan tiempo medio para la falla y tiempo medio para la reparación Sin embargo, el TMEF puede ser problemático por dos razones: 1) proyecta un tiempo entre fallas, pero no da una tasa de fallas proyectada y 2) puede interpretarse mal.

Una medición alternativa de confiabilidad es la de las fallas en el tiempo (FET): medición estadística de cuántas fallas tendrá un componente en mil millones de horas de operación.

Además de una medida de la confiabilidad, también debe desarrollarse otra para la disponibilidad. La disponibilidad del software es la probabilidad de que un programa opere de acuerdo con los requerimientos en un momento determinado de tiempo, y se define así:

Disponibilidad = TMPF / (TMPF + TMPR) \* 100%

Seguridad del software

La seguridad del software es una actividad del aseguramiento del software que se centra en la identificación y evaluación de los peligros potenciales que podrían afectarlo negativamente y que podrían ocasionar que falle todo el sistema. Inicialmente se identifican los peligros y se clasifican según su riesgo. Una vez identificados y analizados los peligros, pueden especificarse requerimientos relacionados con la seguridad para el software. Es decir, la especificación contendría una lista de eventos indeseables y las respuestas deseadas del sistema ante ellos.

LAS NORMAS DE CALIDAD ISO 9000

La norma ISO 9000 describe en términos generales los elementos de aseguramiento de la calidad que se aplican a cualquier negocio. Los requerimientos esbozados por la norma ISO 9001:2000 se dirigen a temas tales como responsabilidad de la administración, sistema de calidad, revisión del contrato, control del diseño, documentación y control de datos, identificación del producto y su seguimiento, control del proceso, inspección y pruebas, acciones correctivas y preventivas, registros del control de calidad, auditorías internas de calidad, capacitación, servicio y técnicas estadísticas.

# MÉTRICAS DE PRODUCTO

MARCO CONCEPTUAL PARA LAS MÉTRICAS DE PRODUCTO

Medidas, métricas e indicadores

una medida proporciona un indicio cuantitativo de la extensión, cantidad, dimensión, capacidad o tamaño de algún atributo de un producto o proceso. La medición es el acto de determinar una medida. El IEEE define métrica como “una medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo determinado”.

Una métrica dde software relaciona medidas individuales.

Un indicador es una métrica o combinación de métricas que proporcionan comprensión acerca del proceso de software.

Principios de medición

Se sugiere un proceso de medición que puede caracterizarse mediante cinco actividades:

• Formulación. La derivación de medidas y métricas de software apropiadas para la representación del software que se está construyendo.

• Recolección. Mecanismo que se usa para acumular datos requeridos para derivar las métricas formuladas.

• Análisis. El cálculo de métricas y la aplicación de herramientas matemáticas.

• Interpretación. Evaluación de las métricas resultantes para comprender la calidad de la representación.

• Retroalimentación. Recomendaciones derivadas de la interpretación de las métricas del producto, transmitidas al equipo de software.

Principios representativos para la caracterizacvion y validación de métricas:

• Una métrica debe tener propiedades matemáticas deseables

• Cuando una métrica representa una característica de software que aumenta cuando ocurren rasgos positivos o que disminuye cuando se encuentran rasgos indeseables

• Cada métrica debe validarse de manera empírica

Medición de software orientado a meta

El paradigma Meta/Pregunta/Métrica (MPM) enfatiza la necesidad de:

1) establecer una meta de medición explícita que sea específica para la actividad del proceso o para la característica del producto que se quiera valorar, 2) definir un conjunto de preguntas que deban responderse con la finalidad de lograr la meta y 3) identificar métricas bien formuladas que ayuden a responder dichas preguntas.

Atributos de las métricas de software efectivas

• Simple y calculable. Debe ser relativamente fácil aprender cómo derivar la métrica y su cálculo no debe demandar esfuerzo o tiempo excesivo.

• Empírica e intuitivamente convincente. Debe satisfacer las nociones intuitivas

• Congruente y objetiva. Siempre debe producir resultados que no tengan ambigüedades.

• Constante en su uso de unidades y dimensiones.

• Independiente del lenguaje de programación.

• Un mecanismo efectivo para retroalimentación de alta calidad.

MÉTRICAS PARA EL MODELO DE REQUERIMIENTOS

Dichas métricas examinan el modelo de requerimientos con la intención de predecir el “tamaño” del sistema resultante.

Métrica basada en funciones

La métrica de punto de función (PF) puede usarse de manera efectiva como medio para medir la funcionalidad que entra a un sistema.4 Al usar datos históricos, la métrica PF puede entonces usarse para: 1) estimar el costo o esfuerzo requerido para diseñar, codificar y probar el software; 2) predecir el número de errores que se encontrarán durante las pruebas, y 3) prever el número de componentes.

Los valores de dominio de información se definen en la forma siguiente:

- Número de entradas externas (EE). Cada entrada externa se origina de un usuario o se transmite desde otra aplicación

- Número de salidas externas (SE). Cada salida externa es datos derivados dentro de la aplicación que ofrecen información al usuario.

- Número de consultas externas (CE). Una consulta externa se define como una entrada en línea que da como resultado la generación de alguna respuesta de software inmediata

- Número de archivos lógicos internos (ALI). Cada archivo lógico interno es un agrupamiento lógico de datos que reside dentro de la frontera de la aplicación

- Número de archivos de interfaz externos (AIE). Cada archivo de interfaz externo es un agrupamiento lógico de datos que reside fuera de la aplicación

MÉTRICAS PARA EL MODELO DE DISEÑO

Métricas del diseño arquitectónico

Las métricas del diseño arquitectónico se enfocan en características de la arquitectura del programa y en la efectividad de los módulos. Para estas se definen tres medidas de complejidad del diseño de software: complejidad estructural, complejidad de datos y complejidad del sistema:

- La complejidad estructural refiere a la estructura jerárquica

- La complejidad de datos ofrece un indicio de la complejidad que hay en la interfaz interna para un módulo.

- la complejidad del sistema se define como la suma de las complejidades estructural y de datos

Métricas para diseño orientado a objetos. Características:

- Tamaño. El tamaño se define en función de cuatro visiones: población, volumen, longitud

y funcionalidad.

- Complejidad: cómo se relacionan mutuamente las clases de un diseño OO.

- Acoplamiento. Las conexiones físicas entre elementos del diseño OO

- Suficiencia. grado en el que una abstracción posee las características requeridas

- Completitud. La completitud considera múltiples puntos de vista, y plantea la pregunta: “¿qué propiedades se requieren para representar por completo al objeto de dominio problema?”

- Cohesion: La cohesividad de una clase se determina al examinar el grado en el que “el conjunto de propiedades que posee es parte del problema

- Primitivismo. es el grado en el que una operación es atómica

- Similitud. El grado en el que dos o más clases son similares

- Volatilidad. mide la probabilidad de que ocurrirá un cambio.

Métricas orientadas a clase: la suite de métricas CK

- Métodos ponderados por clase (MPC). El número de métodos y su complejidad son indicadores razonables de la cantidad de esfuerzo requerido para implementar y probar una clase.

- Profundidad del árbol de herencia (PAH). Esta métrica es “la máxima longitud desde el nodo hasta la raíz del árbol

- Número de hijos (NDH). Las subclases que son inmediatamente subordinadas a una clase en la jerarquía de clase se denominan hijos.

- Acoplamiento entre clases de objetos (ACO). El modelo CRC puede usarse para determinar el valor para el ACO. En esencia, ACO es el número de colaboraciones citadas para una clase en su tarjeta índice CRC

- Respuesta para una clase (RPC). Respuesta para una clase es “un conjunto de métodos que potencialmente pueden ejecutarse en respuesta a un mensaje recibido

- Falta de cohesión en métodos (FCOM). es el número de métodos que acceden a uno o más de los mismos atributos.

Métricas orientadas a clase: La suite de métricas MOOD

- Factor de herencia de método (FHM). El grado en el que la arquitectura de clase de un sistema OO utiliza la herencia tanto para métodos (operaciones) como para atributos

- Factor de acoplamiento (FA). El acoplamiento es un indicio de las conexiones entre elementos del diseño OO.

Métricas de diseño en el nivel de componente

se enfocan en las características internas de un componente de software e incluyen medidas de

cohesión de módulo, acoplamiento y complejidad.

Métricas orientadas a operación

Basadas en las características promedio para los métodos (operaciones). Tres métricas simples, propuestas por Lorenz y Kidd, son apropiadas:

- Tamaño promedio de operación (TOprom). conteo de el número de líneas de código o el de mensajes enviados por la operación.

- Complejidad de la operación (CO).

- Número promedio de parámetros por operación (NPprom).

MÉTRICAS DE DISEÑO PARA WEBAPPS

- Métricas de interfaz.

- Métricas estéticas (diseño gráfico).

- Métricas de contenido. Las métricas en esta categoría se enfocan en la complejidad del contenido y en los grupos de objetos de contenido que se organizan en páginas

- Métricas de navegación. Las métricas en esta categoría abordan la complejidad del flujo de

Navegación.

MÉTRICAS PARA PRUEBAS

Métricas para pruebas orientadas a objetos

- Falta de cohesión en métodos (FCOM)

- Porcentaje público y protegido (PPP).

- Acceso público a miembros de datos (APD).

- Número de clases raíz (NCR).

- Fan-in (FIN): Indica si la jerarquía de herencia es multiple siendo el FIN > 1

- Número de hijos (NDH) y profundidad del árbol de herencia (PAH).

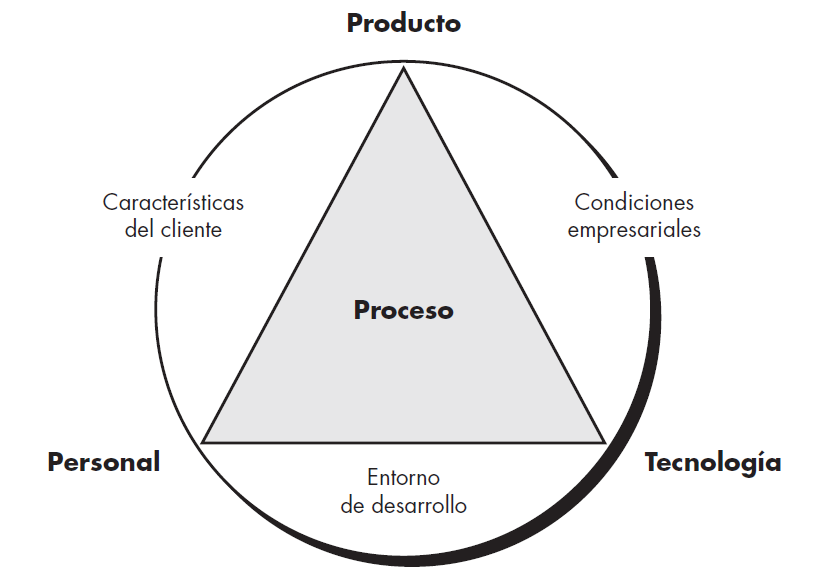
# MÉTRICAS DE PROCESO Y DE PROYECTO

MÉTRICAS EN LOS DOMINIOS DE PROCESO Y PROYECTO

Las métricas de proceso se recopilan a través de todos los proyectos y durante largos espacios de tiempo. Su intención es proporcionar un conjunto de indicadores de proceso que conduzca a mejorar el proceso de software a largo plazo. Las métricas de proyecto permiten al gerente de un proyecto de software: 1) valorar el estado de un proyecto en marcha, 2) rastrear riesgos potenciales, 3) descubrir áreas problema antes de que se vuelvan “críticas”, 4) ajustar el flujo de trabajo o las tareas y 5) evaluar la habilidad del equipo del proyecto para controlar la calidad de los productos operativos del software.

Las métricas del proceso y la mejora del proceso de software

El proceso se asienta en el centro de un triángulo que conecta tres factores, La habilidad y motivación del personal, la complejidad del producto, la tecnología, el circulo de condiciones ambientales que incluyen entorno de desarrollo y características del cliente son factores de influencia sobre la calidad del software.



Métricas de proyecto

Las métricas recopiladas de proyectos anteriores se usan como la base desde la cual se hacen estimaciones de esfuerzo y tiempo para el trabajo de software nuevo. Se miden las tasas de producción representadas en términos de modelos creados, horas de revisión, puntos de función y líneas de fuente entregadas. Además, se rastrean los errores descubiertos durante cada tarea de ingeniería del software.

La intención de las métricas de proyecto es doble. Primero, se usan para minimizar el calendario de desarrollo al hacer los ajustes necesarios para evitar demoras y mitigar potenciales problemas y riesgos. Segundo, se usan para valorar la calidad del producto sobre una base en marcha.

MEDICIÓN DEL SOFTWARE

Métricas orientadas a tamaño

• Errores por KLOC (miles de líneas de código).

• Defectos por KLOC.

• $ por KLOC.

• Páginas de documentación por KLOC.

• Errores por persona-mes.

• KLOC por persona-mes.

• $ por página de documentación.

Métricas orientadas a función

La métrica orientada a función de mayor uso

es el punto de función (PF). El cálculo del punto de función se basa en características del dominio

y de la complejidad de información del software.

Métricas orientadas a objetos

- Número de guiones de escenario. Un guión de escenario (análogo a los casos) es una secuencia detallada de pasos que describen la interacción entre el usuario y la aplicación.

- Número de clases clave. Las clases clave son los “componentes enormemente independientes

- Número de clases de apoyo. Las clases de apoyo se requieren para implementar el sistema, pero no se relacionan de inmediato con el dominio del problema. Los ejemplos pueden ser clases de interfaz de usuario (GUI), clases de acceso y manipulación de base de datos y clases de cálculo.

- Número promedio de clases de apoyo por clase clave.

- Número de subsistemas. Un subsistema es un agregado de clases que apoyan una función que es visible para el usuario final de un sistema.

Métricas orientadas a caso de uso

Los investigadores sugieren puntos de caso de uso (PCU) como un mecanismo para estimar el esfuerzo del proyecto y otras características. Los PCU son una función del número de actores y transacciones implicados por los modelos de caso de uso y su análogo al PF en algunas formas.

Métricas de proyecto webapp

Entre las medidas que pueden recopilarse están (conforme aumenten crece la complejidad):

- Número de páginas web estáticas. Las páginas web con contenido estático (es decir, el usuario final no tiene control sobre el contenido que se despliega en la página).

- Número de páginas web dinámicas. Las páginas web con contenido dinámico (es decir, acciones del usuario final u otros factores externos dan como resultado contenido personalizado que se despliega en la página.

- Número de vínculos de página internos. Los vínculos de página internos son punteros que proporcionan un hipervínculo hacia alguna otra página web dentro de la webapp.

- Número de objetos de datos persistentes. Una webapp puede tener acceso a uno o más objetos de datos persistentes (por ejemplo, una base de datos o archivo de datos).

- Número de sistemas externos puestos en interfaz. Con frecuencia, las webapps deben tener interfaces con aplicaciones empresariales

- Número de objetos de contenido estáticos. Los objetos de contenido estáticos abarcan información basada en texto, gráfica, video, animación y audioestática

- Número de objetos de contenido dinámicos. Los objetos de contenido dinámicos se generan con base en las acciones del usuario final y abarcan información basada en texto, gráfica, video, animación y audio, generada internamente

- Número de funciones ejecutables. Una función ejecutable (por ejemplo, un guión o applet) proporciona cierto servicio computacional al usuario final.

MÉTRICAS PARA CALIDAD DE SOFTWARE

Medición de la calidad

- Exactitud. grado en el cual el software realiza la función requerida.

- Capacidad de mantenimiento. es la facilidad con la que un programa puede corregirse si se encuentra un error, la facilidad con que se adapta si su entorno cambia o de mejorar si el cliente quiere un cambio en requerimientos.

- Integridad. habilidad de un sistema para resistir ataques (tanto accidentales como intencionales) a su seguridad.

- Usabilidad. Cuantificar la facilidad de uso y puede medirse en términos de las características

Eficiencia en la remoción del defecto

En esencia, la ERD es una medida de la habilidad de filtrado de las acciones de aseguramiento y control de la calidad según se aplican a lo largo de todas las actividades del marco conceptual del proceso.

INTEGRACIÓN DE MÉTRICAS DENTRO DEL PROCESO DE SOFTWARE

Argumentos para métricas de software

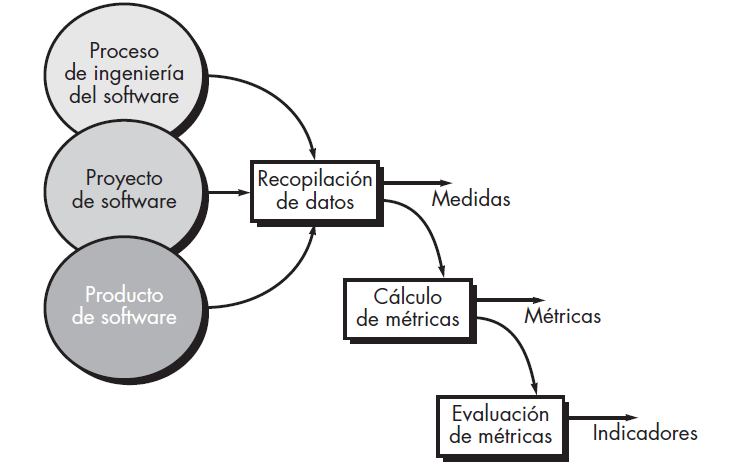
Forma real de determinar si se está mejorando. la medición se utiliza para establecer una línea de referencia del proceso desde el cual puedan valorarse las mejoras.

Establecimiento de una línea de referencia

La línea de referencia de métricas consiste en los datos recopilados a partir de los proyectos de desarrollo

de software. la línea de referencia deben tener los siguientes atributos: 1) deben ser razonablemente precisos y deben evitarse “suposiciones” acerca de los proyectos anteriores, 2) deben recopilarse para tantos proyectos como sea posible, 3) deben ser consistentes (por ejemplo, una línea de código debe interpretarse consistentemente a través de todos los proyectos para los cuales se recopilan datos), 4) las aplicaciones deben ser similares al trabajo que debe estimarse.

Recolección, cálculo y evaluación de métricas



# ADMINISTRACIÓN DEL RIESGO

ESTRATEGIAS REACTIVAS DE RIESGO FRENTE A ESTRATEGIAS PROACTIVAS DE RIESGO

Las estrategias reactivas de riesgo se basan en la reacción ante el problema y no se previene el mismo. Cuando esto sucede, los recursos se hacen a un lado para lidiar con los riesgos, hasta que se convierten en problemas reales.

Una estrategia proactiva comienza mucho antes de iniciar el trabajo técnico. Los riesgos potenciales se identifican, su probabilidad e impacto se valoran y se clasifican por importancia. Luego, el equipo de software establece un plan para gestionar el riesgo. El objetivo principal es evitarlo, pero, dado que no todos los riesgos son evitables, el equipo trabaja para desarrollar un plan de contingencia que le permitirá responder en forma controlada y efectiva.

RIESGOS DE SOFTWARE

Los riesgos siempre involucran dos características: incertidumbre y perdida. Cuando se analizan los riesgos es importante cuantificar el nivel de incertidumbre y el grado de pérdidas asociados con cada riesgo. Para lograr esto, se consideran diferentes categorías de riesgos.

- Los riesgos del proyecto amenazan el plan del proyecto, es decir, si los riesgos del proyecto se vuelven reales, es probable que el calendario del proyecto se deslice y que los costos aumenten.

- Los riesgos técnicos amenazan la calidad y temporalidad del software que se va a producir.

- Los riesgos empresariales amenazan la viabilidad del software que se va a construir y con frecuencia ponen en peligro el proyecto o el producto. Los candidatos para los cinco principales riesgos empresariales son: 1) construir un producto o sistema excelente que realmente no se quiere (riesgo de mercado), 2) construir un producto que ya no encaje en la estrategia empresarial global de la compañía (riesgo estratégico), 3) construir un producto que el equipo de ventas no sabe cómo vender (riesgo de ventas), 4) perder el apoyo de los administradores debido a un cambio en el enfoque o en el personal (riesgo administrativo) y 5) perder apoyo presupuestal o de personal (riesgos presupuestales).

- Los riesgos conocidos son aquellos que pueden descubrirse después de una evaluación cuidadosa del plan del proyecto, del entorno empresarial o técnico donde se desarrolla el proyecto y de otras fuentes de información confiables (por ejemplo, fecha de entrega irreal, falta de requisitos documentados o ámbito de software, pobre entorno de desarrollo). Los riesgos predecibles se extrapolan de la experiencia en proyectos anteriores (por ejemplo, rotación de personal, pobre comunicación con el cliente, disolución del esfuerzo del personal conforme se atienden las solicitudes de mantenimiento). Los riesgos impredecibles son el comodín en la baraja. Pueden ocurrir y lo hacen, pero son extremadamente difíciles de identificar por adelantado.

IDENTIFICACIÓN DE RIESGOS

La identificación de riesgos es un intento sistemático por especificar amenazas al plan del proyecto.

Existen dos tipos distintos de riesgos para cada una de las categorías: riesgos genéricos y riesgos específicos del producto. Los riesgos genéricos son una amenaza potencial a todo proyecto de software. Los riesgos específicos del producto pueden identificarse solamente por quienes tienen clara comprensión de la tecnología, el personal y el entorno específico del software que se construye.

Componentes y promotores de riesgo

• Riesgo de rendimiento: grado de incertidumbre de que el producto satisfará sus requisitos y se ajustará al uso pretendido.

• Riesgo de costo: grado de incertidumbre de que el presupuesto del proyecto se mantendrá.

• Riesgo de apoyo: grado de incertidumbre de que el software resultante será fácil de corregir, adaptar y mejorar.

• Riesgo de calendario: grado de incertidumbre de que el calendario del proyecto se mantendrá y de que el producto se entregará a tiempo.

PROYECCIÓN DEL RIESGO

La proyección del riesgo, también llamada estimación del riesgo, intenta calificar cada riesgo en dos formas: 1) la posibilidad o probabilidad de que el riesgo sea real y 2) las consecuencias de los problemas asociados con el riesgo, en caso de que ocurra. Usted trabaja junto con otros gerentes y personal técnico para realizar cuatro pasos de proyección de riesgo:

1. Establecer una escala que refleje la probabilidad percibida de un riesgo.

2. Delinear las consecuencias del riesgo.

3. Estimar el impacto del riesgo sobre el proyecto y el producto.

4. Valorar la precisión global de la proyección del riesgo de modo que no habrá malos entendidos.

REFINAMIENTO DEL RIESGO

es posible refinar el riesgo en un conjunto de riesgos más detallados, cada uno un poco más sencillo de mitigar, monitorear y manejar. Una forma de hacer esto es representar el riesgo en formato condición-transición-consecuencia. Es decir, el riesgo se enuncia en la forma siguiente:

Dado que <condición> entonces hay preocupación porque (posiblemente) <consecuencia>.

MITIGACIÓN, MONITOREO Y MANEJO DE RIESGO

Una estrategia efectiva debe considerar tres temas: 1) evitar el riesgo, 2) monitorear el riesgo y 3) manejar el riesgo y planificar la contingencia.

Si un equipo de software adopta un enfoque proactivo ante el riesgo, evitarlo siempre es la mejor estrategia. Esto se logra desarrollando un plan para mitigación del riesgo.

Conforme avanza el proyecto, comienzan las actividades de monitoreo de riesgos. El gerente de proyecto monitorea factores que pueden proporcionar un indicio de si el riesgo se vuelve más o menos probable.

El manejo del riesgo y la planificación de contingencia suponen que los esfuerzos de mitigación fracasaron y que el riesgo se convirtió en realidad.

Para un proyecto grande pueden identificarse 30 o 40 riesgos. Si para cada uno se identifican entre tres y siete pasos de manejo de riesgo, ¡el manejo del riesgo puede convertirse en un proyecto por sí mismo! Por esta razón, debe adaptar al riesgo de software la regla de Pareto de 80-20. La experiencia indica que 80 por ciento del riesgo de proyecto global. (es decir, 80 por ciento del potencial para falla del proyecto) puede explicarse por sólo 20 por ciento de los riesgos identificados.

La seguridad del software y el análisis de riesgos son las actividades de aseguramiento de la calidad del software que se enfocan en la identificación y valoración de los riesgos potenciales que pueden afectar al software negativamente y hacer que falle todo un sistema. Si los riesgos pueden identificarse tempranamente en el proceso de ingeniería del software, pueden especificarse características de diseño del software que eliminarán o controlarán los riesgos potenciales.

EL PLAN MMMR

Los pasos de administración del riesgo pueden organizarse en un plan de mitigación, monitoreo y manejo de riesgo (MMMR) por separado. El plan MMMR documenta todo el trabajo realizado como parte del análisis de riesgos y el gerente del proyecto lo usa como parte del plan de proyecto global.

Una vez documentada la MMMR y comenzado el proyecto, inician los pasos de mitigación y monitoreo del riesgo. Como ya se estudió, la mitigación del riesgo es una actividad que busca evitar el problema. El monitoreo del riesgo es una actividad de seguimiento del proyecto con tres objetivos principales: 1) valorar si los riesgos predichos en efecto ocurren, 2) asegurar que los pasos para evitar el riesgo definidos para un riesgo determinado se aplican de manera correcta y 3) recopilar información que pueda usarse para futuros análisis de riesgos.