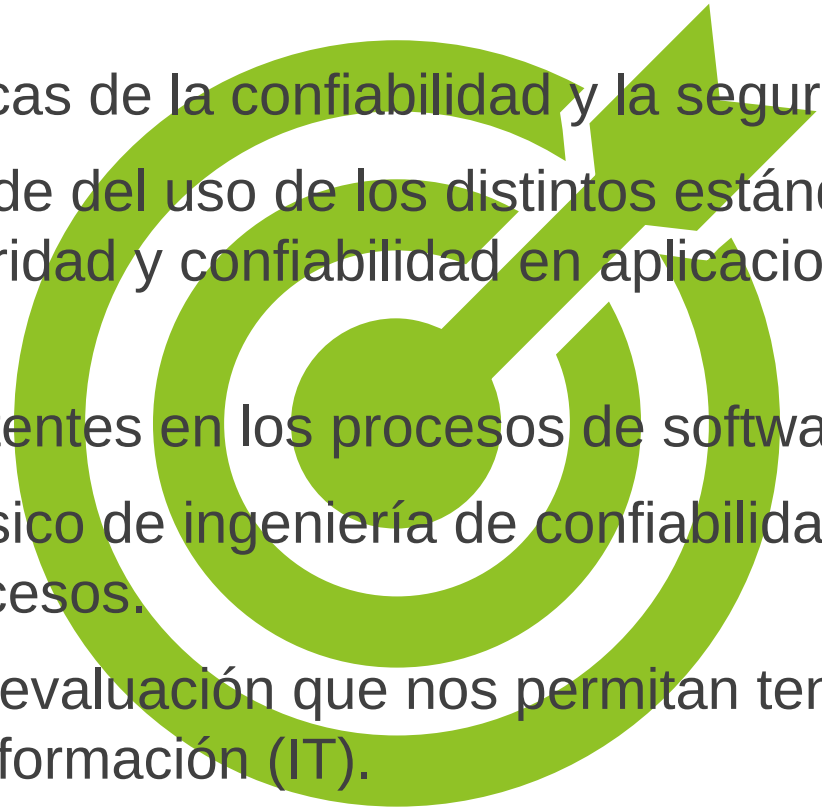
TÓPICOS AVANZADOS DE INGENIERÍA DE SOFTWARE

MG. DELY MARYSHECK LAZO BARREDA

CONFIABILIDAD Y SEGURIDAD EN EL SOFTWARE

Objetivos

- Conocer las características de la confiabilidad y la seguridad en software.
- Trasmitir la importancia de del uso de los distintos estándares, métodos y procedimientos de seguridad y confiabilidad en aplicaciones, paginas web y sistemas.
- Identificar las fallas existentes en los procesos de software.
- Implementar un plan básico de ingeniería de confiabilidad, en base a fallas encontradas en sus procesos.
- Conocer los criterios de evaluación que nos permitan tener la certeza de la confidencialidad de la Información (IT).
- Conocer los conceptos básicos para mantener la seguridad dentro de los sitios de la Internet.



- Temas a tratar
  - Ingeniería de Confiabilidad.
  - Confiabilidad de Software.
  - Seguridad de Software.
  - Riesgos de seguridad.
  - Vulnerabilidades.

Introducción

- Los sistemas de información son cada vez más importantes en la sociedad moderna.
- A medida que las funciones de este tipo de sistemas son más esenciales y complejas, su confiabilidad es más crítica e importante.
- A raíz de todo esto el software inseguro está debilitando las finanzas, salud, defensa, energía, y otras infraestructuras críticas.
- A medida que la infraestructura digital se hace cada vez más compleja e interconectada, la dificultad de lograr la seguridad en aplicaciones aumenta exponencialmente.
- Es por tal motivo que hoy en día muchas de las organizaciones tan tomando medidas para garantizar la seguridad del software así como su valor consecuente como es la confiabilidad, es por ende que hoy en día es muy importante para todos la medición de la calidad de sw y la implementación de dos características muy importantes al momento de desarrollar, como son la confiabilidad y la seguridad.

Calidad

- El son de calidad mejorada del software comenzó en serio a medida que el software se integraba cada vez mas en todas las facetas de nuestras vidas. Para la década de 1990, las principales corporaciones reconocieron que se desperdiciaban miles de millones de dólares al año en software que no ofrecía las características y la funcionalidad que habían prometido peor aún, tanto al gobierno como la industria les comenzó a preocupar cada vez mas el hecho de una falla importante del software pudiera paralizar infraestructura importante provocado costos de decenas de miles de millones más.
- Al concluir el siglo, CIO Magazine proclamó el siguiente encabezado: “Dejemos de desperdiciar 78 mil millones al año” lamentando el hecho de que las empresas estadounidense gastaran miles de millones en software que no hacia lo que debería”. Por desgracia, al menos una de las encuestas sobre el estado de la calidad del software conforman hasta el 90% de los costos totales de desarrollo del software. La mala calidad del software provocada por la prisa de liberar productos sin pruebas adecuadas sigue atormentado a la industria del software.

ANÁLISIS BREVE

¿Qué es? La respuesta no es tan sencilla como podríamos pensar. Sabemos identificar la calidad cuando la vemos y, aun así, puede llegar a ser difícil de definir. Pero para el software de computadora, la calidad es algo que debemos definir y es lo que haremos en este capítulo.

¿Quién lo hace? Todos (ingenieros de software, gerentes, todas las partes interesadas) los involucrados en el proceso del software son responsables de la calidad.

¿Por qué es importante? Podemos hacerlo bien, o podemos hacerlo de nuevo. Si un equipo de software hace hincapié en la calidad en todas las actividades de ingeniería de software, reduce la cantidad de reelaboración que puede existir. Esto provoca menores costos y, lo que es más importante, el lanzamiento a un mejor plazo.

¿Cuáles son los pasos? Para lograr software de alta calidad, deben ocurrir cuatro actividades: un proceso y práctica de ingeniería de software comprobados, gestión sólida del proyecto, control de calidad exhaustivo y la presencia de una infraestructura de aseguramiento de calidad.

¿Qué es el producto de trabajo? Software que cumple con las necesidades de sus clientes, que se desempeña en forma precisa y confiable, y que provee valor para todos los que lo usan.

¿Cómo me aseguro de que lo hice bien? Se rastrea la calidad mediante un análisis de los resultados de todas las actividades de control de calidad y se mide mediante un análisis de los errores antes de la entrega y los defectos que se liberan al campo.

CALIDAD

- En la actualidad la calidad del software sigue siendo un problema, pero ¿quien tiene la culpa? los clientes culpan a los desarrolladores, argumentado que las practicas descuidas conducen a software de baja calidad. Los desarrolladores culpan a los clientes(y a otras partes interesadas), argumentado que las fechas de entrega irracionales y un flujo continuo de cambios los obliga a entregar software antes de que se haya validado por completo. ¿Quién tiene la razón? Ambos, y ese es el problema.

¿QUÉ ES LA CALIDAD?

- A un nivel un poco mas pragmático. David Garvin de la Harvard Business School sugiere que “la calidad es un concepto complejo y multifacético” que puede describirse desde cinco puntos de vistas diferentes. La vista trascendental alega que la calidad es algo que uno reconoce de inmediato pero que no puede definir de manera explicita. La vista del usuario ve la calidad en términos de las metas especificas de un usuario final si un producto cumple esas metas, exhibe calidad. La vista del fabricante define la calidad en términos de la especificación original del producto. Si este se conforma a la especificación, exhibe calidad. La vista del producto sugiere que la calidad puede enlazarse a las características inherentes(por ejemplo, funciones y rasgos) de un producto. Por ultimo, la vista basada en el valor mide la calidad con base en cuanto esta dispuesto apagar un cliente por producto.

Calidad del diseño

- Se refiere a las características que los diseñadores especifican para un producto. El tipo de materiales, tolerancias y especificaciones de rendimiento contribuyen para la calidad del diseño. A medida que se usan mejores materiales se especifican tolerancias mas altas y mayores niveles de rendimiento, y la calidad de un producto aumenta si este se fabrica de acuerdo con las especificaciones.
- En el desarrollo de software, la calidad del diseño abarca el grado con el que el diseño cumple las funciones y características especificas en el modelo de requerimientos. La calidad de conformidad se enfoca en el grado con el que la implementación sigue el diseño y el sistema resultante cumple con sus requerimientos y metas de rendimiento.

Satisfacción del usuario= producto funcional+ buena calidad + entrega a tiempo y dentro del presupuesto

Calidad

- En conclusión. Glass afirma que la calidad es importante, pero si el usuario no esta satisfecho, en realidad no importa nada mas, DeMarco refuerza esta opinión al decir “La calidad de un producto es una función de que tanto cambia el mundo para mejorarlo”. Esta vista de la calidad sostiene que, si un producto de software provee un beneficio considerable o rendimiento. Una vista moderada de la calidad del software requiere atención en la satisfacción del cliente, así como el cumplimiento de los requerimientos del producto.

Calidad del software

- Incluso hasta los desarrolladores de software más experimentados estarán de acuerdo en que el software de alta calidad es una meta importante. Pero ¿ cómo definimos la calidad del software? podemos definir la calidad del software así; un proceso de software efectivo, aplicado de una forma que cree un producto útil, el cual ofrezca un valor cuantificable para quienes lo producen y quienes lo utilizan.
- La definición sirve para hacer énfasis en tres puntos importantes:
  - Un proceso de software efectivo establece la infraestructura que apoya cualquier esfuerzo de crear un producto de software de alta calidad. Los aspectos administrativos del proceso crean la verificaciones y cálculos que ayudan a evitar un caos en el proyecto un contribuyente clave de la mala calidad. Las practicas de ingeniería de software permiten al desarrollador analizar el problema y diseñar una solución solida: ambas cosas son imprescindibles para crear software del alta calidad. Por ultimo, las actividades paraguas tales como la gestión del cambio y las revisiones técnicas tienen tanto que ver con la calidad como cualquier otra parte de la practica de ingeniería de software.
  - Un producto útil entrega el contenido, las funciones y las características que el usuario final desea, pero también es importante que entregue estos activos de una forma confiable y libre de errores un producto útil siempre satisface los requerimientos que las partes interesadas han declarado de manera explicita. Además satisface un conjunto de requerimientos implícitos(como facilidad de uso) que se esperan de todo el software de alta calidad.

Calidad del software

- La definición sirve para hacer énfasis en estos puntos importantes:
  - Al agregar valor para el productor y el usuario de un producto de software de alta calidad provee beneficios para organización y la comunidad del usuarios finales. La organización de software obtiene un valor adicional, ya que software de alta calidad requiere menos mantenimiento y esfuerzo, menos correcciones de errores y un soporte reducido para el cliente.
  - Esto permite a los ingenieros de software invertir mas tiempo en la creación de nuevas aplicaciones y menos en la reelaboración. La comunidad de usuarios obtiene valor agregada, ya que la aplicación provee una capacidad útil de forma que agiliza cierto proceso de negocios. El resultado final es(1) mas ingresos por el producto de software. (2) mayor rentabilidad cuando una aplicación apoya un proceso de negocio/o (3) mayor disponibilidad de la información que es crucial para el negocio.

Factores de calidad del software

- McCall y Walters propusieron una forma útil de pensar en, y organizar los factores que afectan a la calidad del software. Sus factores de calidad del software sus factores de calidad del software se enfocan en tres aspectos del producto de software: sus características de operación, su habilidad de someterse al cambio y su capacidad de adaptarse a nuevos entornos.

Factores de calidad del software

El modelo de calidad del producto describe ocho características que se enfocan en las naturalezas dinámica y estática de los sistemas de computadora.

Modelo de calidad en el uso

- Efectividad. La precisión y exactitud con que los usuarios logran las metas.
- Eficiencia. Los recursos que se gastan para lograr las metas del usuario completamente y con la precisión deseada.
- Satisfacción. Utilidad confianza, placer y comunidad
- Liberación del riesgo. Mitigación de los riesgos económicos, de la salud de seguridad y ambientales.
- Cobertura del contexto. Plenitud y flexibilidad.

Calidad del producto

- Idoncidad funcional. Completo correcto y apropiado.
- Eficiencia en el rendimiento. Sincronización, uso de recursos y capacidad.
- Compatibilidad. Coexistencia e interoperabilidad.
- Facilidad de uso. Adecuación, facilidad de aprendizaje, operabilidad, protección de errores, estática y accesibilidad.
- Confiabilidad. Madurez, disponibilidad, tolerancia a fallas y capacidad de recuperación.
- Seguridad. Confidencialidad, integridad, rendición de cuentas y autenticidad.
- Facilidad de mantenimiento. modularidad, reutilización, capacidad de modificación y facilidad de prueba.
- Portabilidad. Adaptabilidad, capacidad de instalación y facilidad de reemplazo.

Evaluación de la calidad cualitativa

¿Qué tan rápido pueden determinar los usuarios si el producto de software puede usarse para ayudarlos a completar sus tareas o no? (adecuación)

¿Cuánto tiempo toma a los usuarios aprender a usar las funciones del sistema necesarias para completar su tarea? (facilidad de aprendizaje)

¿Puede el usuario recordar cómo usar las funciones del sistema en sesiones posteriores de prueba sin tener que volver a aprenderlas? (facilidad de aprendizaje)

¿Cuánto tiempo tardan los usuarios en completar las tareas usando el sistema? (operabilidad)

¿Trata el sistema de evitar que los usuarios cometan errores? (protección de errores)

¿Permite el sistema que los usuarios cometan errores? (protección de errores)

¿Las respuestas contestan de manera favorable las preguntas sobre la apariencia de la interfaz de usuario? (estética)

¿Se conforma la interfaz a las expectativas establecidas por las reglas doradas? (accesibilidad)

¿Se conforma la interfaz de usuario a los elementos de la lista de comprobación de accesibilidad requeridos para los usuarios previstos? (accesibilidad)

Evaluación de la calidad cuantitativa

- Es posible detectar varios defectos de diseño de software mediante la métrica de software.
- El proceso consiste en buscar fragmentos de código que sugieran la presencia de cosas como un acoplamiento elevado o niveles innecesarios de complejidad. Los atributos de código internos pueden describirse de manera cuantitativa mediante la métrica de software. Siempre que los valores de la métrica del software calculados para un fragmento de código estén fuera del rango de valores aceptables, se indica la existencia de un problema de calidad que debe investigarse.

Costo de reparación de software

CasaSegura

Cuestiones de calidad

El escenario: Oficina de Doug Miller mientras empieza el proyecto de software *CasaSegura*.

Los participantes: Doug Miller, gerente del equipo de ingeniería de software de *CasaSegura* y otros miembros del equipo de ingeniería de software del producto.

La conversación:

Doug: Estaba viendo un informe industrial sobre los costos de reparación de defectos en el software. Es muy ilustrativo.

Jamie: Ya estamos trabajando en el desarrollo de casos de prueba para cada requerimiento funcional.

Doug: Eso es bueno, pero me di cuenta de que cuesta ocho veces más reparar un defecto que se descubre en la prueba que si dicho defecto se detecta y repara durante la codificación.

Vinod: Estamos usando programación por pares, así que deberíamos ser capaces de detectar la mayoría de los defectos durante la codificación.

Doug: Creo que no has entendido bien. La calidad es más que el simple hecho de eliminar errores de codificación. Necesitamos analizar las metas de calidad del proyecto y asegurar que los productos de software evolutivos las cumplan.

Jamie: ¿Te refieres a cosas como facilidad de uso, seguridad y confiabilidad?

Doug: Es correcto. Necesitamos crear verificaciones en el proceso de software para monitorear nuestro progreso en cuanto al cumplimiento de nuestras metas de calidad.

Vinod: ¿No podemos terminar el primer prototipo y luego comprobar su calidad?

Doug: Me temo que no. Debemos establecer una cultura de calidad desde las primeras etapas del proyecto.

Vinod: ¿Qué quieres que hagamos, Doug?

Doug: Creo que tendremos que buscar una técnica que nos permita monitorear la calidad de los productos de *CasaSegura*. Pensemos en esto y volvamos a tocar el tema mañana.

Riesgos

- Durante el mes de noviembre del año 2000 en un hospital en Panamá, 28 pacientes recibieron dosis masivas de rayos gama durante su tratamiento para una variedad de cánceres. En los primeros meses, 5 de esos pacientes murieron de envenenamiento por radiación y otros 15 desarrollaron complicaciones graves. ¿Qué fue lo que provocó esta tragedia?
- Los técnicos del hospital modificaron un paquete de software, desarrollado por una empresa estadounidense, para calcular dosis modificadas de radiación para cada paciente. Los tres físicos médicos panameños, que ajustaron el software para proveer capacidad adicional, fueron acusados de homicidio en segundo grado. La empresa estadounidense enfrentó litigios serios en dos países.
- Gage y McCornick comentan: Este no es un cuento aleccionador para técnicos médicos, a pesar de que tal vez tengan que luchar por no ser encarcelados si no entienden o hacen mal uso de la tecnología. Esto tampoco es un cuento acerca de cómo los seres humanos pueden lesionarse o incluso morir debido a un software mal diseñado o explicado, aunque hay suficientes ejemplos para demostrarlo. Esta es una advertencia para los creadores de programas de computadora: que la calidad del software importa, que las aplicaciones deben ser a prueba de fallas y que el código mal desplegado puede matar.

Revisiones

Análisis breve

¿Qué es? A medida que desarrolle productos de trabajo de ingeniería de software se cometerán errores. No hay por qué avergonzarse de ello, siempre y cuando se esfuerce mucho por encontrar y corregir los errores antes de entregar el producto a los usuarios finales. Las revisiones técnicas son el mecanismo más efectivo para encontrar errores en las primeras etapas del proceso del software.

¿Quién lo hace? Los ingenieros de software realizan las pruebas técnicas, también conocidas como revisiones de pares, con sus colegas. Como vimos en los capítulos 3 y 4, algunas veces conviene incluir a otras partes interesadas en estas revisiones.

¿Por qué es importante? Si encuentra un error al principio del proceso, es menos costoso corregirlo. Además, los errores tienen una forma de amplificarse a medida que avanza el proceso. Por lo tanto, un error relativamente menor que no se resuelva en las primeras etapas del proceso puede amplificarse y convertirse en un conjunto importante de errores en etapas posteriores del proyecto. Por último, las revisiones ahorran tiempo al reducir la cantidad de reelaboración requerida posteriormente en el proyecto.

¿Cuáles son los pasos? Su enfoque con respecto a las revisiones variará dependiendo del tipo de revisión que seleccione. En general se emplean seis pasos, aunque no todos se usan para todos los tipos de revisiones: planear, preparar, estructurar la reunión, señalar errores, realización de correcciones (se hace fuera de la revisión) y verificar que las correcciones se hayan realizado de manera apropiada.

¿Qué es el producto de trabajo? El resultado de una revisión es una lista de los problemas y/o errores que se descubrieron. Además, también se indica el estado técnico del producto de trabajo.

¿Cómo me aseguro de que lo hice bien? Primero, seleccione el tipo de revisión apropiado para su cultura de desarrollo. Siga los lineamientos que conduzcan a revisiones exitosas. Si las revisiones que realice producen software de alta calidad, lo hizo bien.

Impacto del costo de los defectos del software

INFORMACIÓN

Bugs, errores y defectos

La meta del control de calidad del software y, en un sentido más amplio, la gestión de la calidad en general es eliminar los problemas de calidad en el software. Estos problemas se conocen por diversos nombres: *bugs*, *fallas*, *errores* o *defectos*, por nombrar algunos. ¿Son estos términos sinónimos, o hay diferencias sutiles entre ellos?

En este libro hacemos una clara distinción entre un *error* (un problema de calidad que se descubre *antes* de liberar el software a otras partes interesadas o usuarios finales) y un *defecto* (un problema de calidad que se descubre solo *después* de haber liberado el software a los usuarios finales u otras partes interesadas).[1] Hacemos esta distinción ya que los errores y los defectos tienen impactos económicos, de negocios, psicológicos y humanos muy diferentes. Como ingenieros de software, queremos encontrar y corregir tantos errores como sea posible antes de que el cliente y/o el usuario final los encuentren.

Queremos evitar los defectos, ya que estos (lo que es justificable) hacen ver mal a las personas encargadas del software.

Es importante señalar, sin embargo, que la distinción temporal entre errores y defectos en este libro *no* es un pensamiento prevalente. El consenso general dentro de la comunidad de ingeniería de software es que los defectos y errores, fallas y bugs son sinónimos. Es decir, el punto en el tiempo en el que se encontró el problema no influye en el término usado para describir el problema. Parte del argumento a favor de esta opinión es que algunas veces es difícil hacer una distinción clara entre previo y posterior a la liberación (por ejemplo, considere un proceso incremental utilizado en el desarrollo ágil).

Sin importar cómo elija interpretar estos términos, es importante reconocer que el punto en el tiempo en el que se descubre un problema es importante y que los ingenieros de software deben esforzarse (mucho) por encontrar los problemas antes de que sus clientes y los usuarios finales los encuentren.

Amplificación y eliminación de defectos

- Ayuda a justificar el esfuerzo que se gasta en las revisiones de software. En esencia, la amplificación de defectos tiene el siguiente argumento: un error que se introduce al comienzo del flujo de trabajo de ingeniería de software (por ejemplo, durante el modelado de requerimientos) y no se detecta, puede y a menudo se amplificará en varios errores durante el diseño. Si esos errores no se descubren (mediante revisiones efectivas), pueden llegar a amplificarse en todavía más errores durante la codificación. Un solo error introducido al principio, que no se descubra y corrija, puede amplificarse en varios errores posteriormente en el proceso. La propagación de defectos es un término que se usa para describir el impacto que tiene un error no descubierto en las actividades de desarrollo futuras o en el comportamiento del producto.

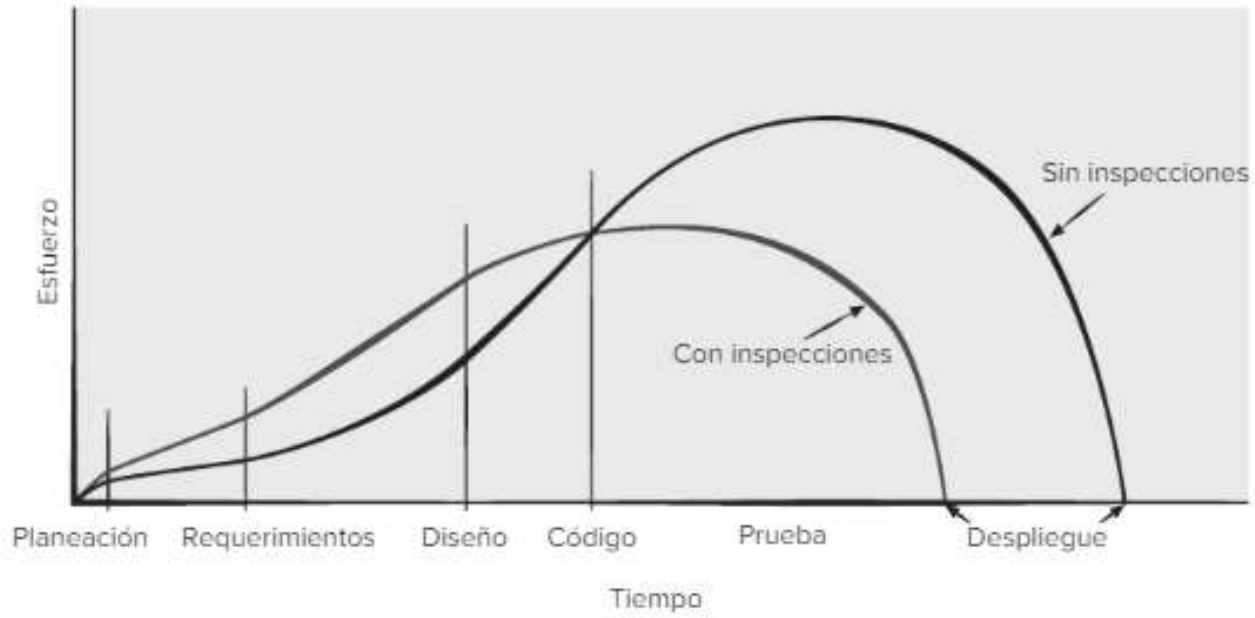
Métricas de revisión y su uso

- Las revisiones técnicas son una de varias acciones que se requieren como parte de la buena práctica de ingeniería de software. Cada acción requiere de esfuerzo humano dedicado. Debido a que el esfuerzo disponible para el proyecto es finito, es importante que una organización de ingeniería de software entienda la efectividad de cada acción mediante la definición de un conjunto de métricas que pueden usarse para evaluar su eficacia.
- Las siguientes métricas de revisión pueden recolectarse para cada revisión que se realice:
  - Esfuerzo de preparación.
  - Esfuerzo de evaluación.
  - Esfuerzo de reelaboración.
  - Esfuerzo de revisión.
  - Tamaño del producto de trabajo.
  - Errores menores encontrados.
  - Errores mayores encontrados.
  - Total de errores encontrados.
  - Densidad de errores encontradas.

Esfuerzo invertido con y sin revisiones

Esfuerzo invertido con y sin revisiones

Fuente: Fagan, Michael E., "Advances in Software Inspections", IEEE Transactions on Software Engineering, vol. SE-12 núm. 7, julio de 1986, 744-751.



Revisiones técnicas formales

CasaSegura

Aspectos de calidad

El escenario: Oficina de Doug Miller, mientras comienza el proyecto de *CasaSegura*.

Los participantes: Doug Miller, gerente del equipo de ingeniería de software de *CasaSegura* y otros miembros del equipo de ingeniería de software del producto.

La conversación:

Doug: Sabemos que no invertimos tiempo en desarrollar un plan de calidad para este proyecto, pero ya estamos en ello y tenemos que considerar la calidad, ¿cierto?

Jamie: Seguro. Ya decidimos que, mientras desarrollamos el modelo de requerimientos [capítulo 8], Ed se comprometió a desarrollar un procedimiento de prueba para cada requerimiento.

Doug: Eso está muy bien, pero no vamos a esperar hasta la prueba para evaluar la calidad, ¿o sí?

Vinod: ¡No! Desde luego que no. Programamos revisiones en el plan del proyecto para este incremento de software. Comenzaremos el control de calidad con las revisiones.

Jamie: Me preocupa un poco el hecho de que no tengamos suficiente tiempo para realizar todas las revisiones. De hecho, estoy segura de que no podremos.

Doug: Mmm. Entonces ¿qué propones?

Jamie: Digo que seleccionemos los elementos de los modelos de requerimientos y de diseño que sean más imprescindibles para *CasaSegura* y los revisemos.

Vinod: Pero ¿qué hay si nos falta algo en una parte del modelo que no revisemos?

Jamie: Puede ser... pero no estoy segura de que tengamos tiempo para revisar cada elemento de los modelos.

Vinod: ¿Qué quieres que hagamos, Doug?

Doug: Tomemos algo de la programación extrema [capítulo 3]. Desarrollaremos los elementos de cada modelo en pares (dos personas) y realizaremos una revisión informal de cada uno a medida que avancemos. Luego nos enfocaremos en los elementos "imprescindibles" para una revisión más formal en equipo, pero mantendremos esas revisiones en un número mínimo. De esa forma, varias personas podrán analizar todo y seguiremos manteniendo nuestras fechas de entrega.

Jamie: Eso significa que tendremos que revisar el calendario.

Doug: Que así sea. La calidad es más importante que el calendario en este proyecto.

Evaluaciones Postmortem

- Pueden aprenderse muchas lecciones si un equipo de software se toma el tiempo de evaluar los resultados de un proyecto después de entregarlo a los usuarios finales. Baaz y sus colegas sugieren el uso de una evaluación Postmortem (EPM) como mecanismos para determinar lo que salió bien y mal a la hora de aplicar el proceso y la práctica de ingeniería de software en un proyecto específico.
- Una EMP estudia todo el proyecto de software y se enfoca en las excelencias (es decir, los logros y las experiencias positivas) y los desafíos (problemas y experiencias negativas).

Revisiones ágiles

- Durante la reunión de planificación de sprint, se revisan las historias de usuario y se ordenan de acuerdo con su prioridad, antes de seleccionar las historias de usuario a incluir en el siguiente sprint. La reunión diaria de Scrum es una forma informal de asegurar que todos los miembros del equipo esten trabajando en las mismas prioridades y detectar los defectos que puedan evitar que se complete el sprint a tiempo.

Resumen

- El propósito de toda revisión técnica es encontrar errores y descubrir aspectos que pudieran tener un impacto negativo en el software que se va a desarrollar. Entre más pronto se descubra un error y se corrija, menos probable será que dicho error se propague a otros productos de trabajo de ingeniería de software y se amplifique, lo que provocará que se requiera un esfuerzo mucho más considerable para corregirlo.
- Para determinar si las actividades de control de calidad están funcionando o no, un conjunto de métricas de revisión deben ser recolectadas. Estas métricas se enfocan en el esfuerzo requerido para realizar la revisión y los tipos y la gravedad de los errores descubiertos durante la revisión. Una vez que se recolectan los datos de las métricas, pueden usarse para evaluar la eficacia de las revisiones que realizamos. Los datos de la industria indican que las revisiones proveen un retorno considerable sobre la inversión.

Aseguramiento de la calidad del software

ANÁLISIS BREVE

¿Qué es? No basta con decir que la calidad del software es importante; es necesario (1) definir de manera explícita lo que queremos decir con "calidad del software", (2) crear un conjunto de actividades que ayuden a garantizar que todo producto de trabajo de ingeniería de software exhiba una alta calidad, (3) realizar actividades de control y aseguramiento de calidad en todos los proyectos de software, (4) usar métricas para desarrollar estrategias que mejoren nuestro proceso del software y, como consecuencia, la calidad del producto final.

¿Quién lo hace? Todos los involucrados en el proceso de ingeniería de software son responsables de la calidad.

¿Por qué es importante? Podemos hacerlo bien, o hacerlo de nuevo. Si un equipo de software hace hincapié en la calidad en todas las actividades de ingeniería de software, reduce la cantidad de reelaboración que puede existir. Esto da como resultado menores costos y, lo que es más importante, una mejora en el tiempo de comercialización.

¿Cuáles son los pasos? Antes de poder iniciar las actividades de aseguramiento de la *calidad del software* (ACS), es importante definir el significado en diversos niveles de abstracción. Una vez que entienda lo que es la calidad, un equipo de software debe identificar un conjunto de actividades de ACS que filtren los errores de los productos de trabajo antes de aprobarlos.

¿Qué es el producto de trabajo? Se crea un plan de aseguramiento de la calidad del software para definir la estrategia de ACS del equipo de software. Durante el modelado y la codificación, el producto de trabajo de ACS es el resultado de las revisiones técnicas (capítulo 16). Durante las pruebas (capítulos 19 al 21), se producen los planes y procedimientos de prueba. También pueden generarse otros productos de trabajo asociados con la mejora del proceso.

¿Cómo me aseguro de que lo hice bien? ¡Encuentre los errores antes de que se conviertan en defectos! Esto es, trabaje para mejorar su eficiencia en la eliminación de defectos (capítulo 23), reduciendo por lo tanto la cantidad de reelaboración que su equipo de software debe realizar.

Aseguramiento de calidad de software

- ACS comprende: un proceso de ACS, tareas específicas de aseguramiento y control de la calidad, una práctica efectiva de ingeniería de software (métodos y herramientas), control de todos los productos de trabajo de software y los cambios realizados n ellos, un procedimiento para asegurar el cumplimiento con los estándares de desarrollo de software y mecanismo de

Elementos de ACS

- Estándares: El IEEE, la ISO y otras organizaciones de estándares han producido una amplia gama de estándares para ingeniería de software y documentos relacionados. Estos pueden ser adoptados en forma voluntaria por una organización de ingeniería de software o impuestos por el cliente u otras partes interesadas.
- Revisiones y auditorias: Las revisiones técnicas son una actividad de control de calidad realizada por los ingenieros de software para otros ingenieros de software. SU intención es descubrir errores.
- Prueba: es una función de control de calidad con una meta principal: encontrar errores. El trabajo del ACS es asegurar que la prueba se planee de manera apropiada y se lleve a cabo en forma eficiente, de modo que tenga la mayor probabilidad de lograr su meta principal.

Elementos de ACS

- Recolección y análisis de errores/defectos: La única forma de mejorar es medir nuestro progreso. El ACS recolecta y analiza los datos de errores y defectos para entender mejor cómo se cometen los errores y qué actividades de ingeniería de software son más adecuadas para eliminarlos.
- Gestión del cambio: El cambio es uno de los aspectos más molestos de cualquier proyecto de software. Si no se gestiona de manera adecuada, puede provocar confusión y esto casi siempre conduce a una mala calidad. El ACS se asegura de que se hayan instituido las prácticas de gestión del cambio adecuadas.
- Educación: Toda organización de software desea mejorar sus prácticas de ingeniería de software. Un colaborador clave para la mejora es la educación de los ingenieros de software, sus gerentes y otras partes interesadas.

Elementos de ACS

- Gestión de seguridad: con el incremento de la ciberdelincuencia y las nuevas regulaciones gubernamentales con respecto a la privacidad, toda organización de software debe instituir políticas que protejas datos en todos los niveles, establecer protección mediante cortafuegos para aplicaciones móviles y asegurar que el software no se haya alterado desde el interior.
- Seguridad: Puesto que el software es casi siempre un componente fundamental de los sistemas clasificados para humanos (como las aplicaciones automotrices o de aeronáutica), el impacto de los defectos ocultos pueden ser catastrófico. El ACS puede ser responsable de evaluar el impacto de la falta del software y de iniciar los pasos requeridos para reducir el riesgo.
- Gestión del riesgo: Aunque el análisis y la mitigación del riesgo son del interés de los ingenieros de software, la organización de ACS se asegura de que las actividades de gestión del riesgo se realicen de manera apropiada y que se hayan establecido planes de contingencia relacionados con el riesgo.

Tareas, metas y métricas de ACS

- Tareas de ACS: La instrucción del grupo de ACS es ayudar al equipo de software a lograr un producto final de alta calidad. El Instituto de Ingeniería de Software recomienda un conjunto de actividades de ACS que aborden la planeación, supervisión, mantenimiento de registros, análisis y generación de informes de aseguramiento de calidad.
  - Prepara un plan ACS para un proyecto.
  - Participa en el desarrollo de la descripción del proceso de software del proyecto.
  - Revisa las actividades de ingeniería de software para verificar el cumplimiento con el proceso de software definido.
  - Audita los productos de trabajo de software designados para verificar el cumplimiento con los que se definen como parte del proceso de software.
  - Se asegura de que las desviaciones en el trabajo del software y los productos de trabajo se documentes y manipulen de acuerdo con un procedimiento documentado.
  - Registra cualquier incumplimiento y lo reporta a la gerencia de nivel superior.

Tareas, metas y métricas de ACS

Tareas, metas y métricas de ACS

CASASEGURA

Aseguramiento de la calidad del software

El escenario: Oficina de Doug Miller, al comenzar el proyecto de software *CasaSegura*.

Los participantes: Doug Miller, gerente del equipo de ingeniería de software de *CasaSegura* y otros miembros del equipo de ingeniería del producto de software.

La conversación:

Doug: ¿Cómo van las cosas con las revisiones informales?

Jamie: Estamos realizando revisiones informales de los elementos críticos del proyecto en pares a medida que codificamos, pero antes de la prueba. Va más rápido de lo que pensaba.

Doug: Eso es bueno, pero quiero que el grupo de ACS de Bridget Thorton realice auditorías de nuestros productos de trabajo para asegurar que sigamos nuestros procesos y cumplamos con nuestras metas de calidad.

Vinod: ¿No están haciendo ya la mayor parte de las pruebas?

Doug: Es cierto. Pero el aseguramiento de la calidad (AC) es más que probar. Necesitamos estar seguros de que nuestros documentos evolucionan junto con nuestro código y de no introducir errores al integrar nuevos componentes.

Jamie: En realidad no quiero que me evalúen con base en sus resultados.

Doug: No te preocupes. Las auditorías se enfocan en el cumplimiento de nuestros productos de trabajo con los requerimientos y las actividades del proceso que definimos. Solo usaremos los resultados de la auditoría para tratar de mejorar nuestros procesos, así como nuestros productos de software.

Vinod: Pues creo que se va a ocupar más de nuestro tiempo.

Doug: A la larga nos ahorrará tiempo cuando encontremos los defectos lo antes posible. Además, cuesta menos corregir los defectos si se detectan de manera anticipada.

Jamie: Suena bien entonces.

Doug: También es importante identificar las actividades en donde se introdujeron los defectos y agregar tareas de revisión para detectarlos en el futuro.

Vinod: Eso nos ayudará a determinar si estamos muestreando con el suficiente cuidado con nuestras actividades de revisión.

Doug: Creo que las actividades de ACS nos convertirán en un mejor equipo a la larga.

Metas, atributos y métricas

- Las actividades de ACS se realizan para lograr un conjunto de metas pragmáticas:
  - Calidad de los requerimientos.
  - Calidad del diseño.
  - Calidad del código.
  - Efectividad del control de calidad.

Metas, atributos y métricas

Meta	Atributo	Métrica
Calidad del requerimiento	Ambigüedad	Número de modificadores ambiguos (como varios, grande, amigable para el humano)
	Integridad	Número de TBA, TBD
	Facilidad de entendimiento	Número de secciones/subsecciones
	Volatilidad	Número de cambios por requerimiento
		Tiempo (por actividad) cuando se solicita un cambio
	Facilidad de rastreo	Número de requerimientos que no se pueden rastrear hasta el diseño/código
	Claridad del modelo	Número de modelos de UML
		Número de páginas descriptivas por modelo
		Número de errores de UML
Calidad del diseño	Integridad de arquitectura	Existencia de un modelo arquitectónico
	Integridad del componente	Número de componentes que se rastrean hasta el modelo de arquitectura
		Complejidad del diseño procedimental
	Complejidad de la interfaz	Número promedio de selecciones para llegar a una función o contenido típicos
		Adecuación del esquema
	Patrones	Número de patrones utilizados
Calidad del código	Complejidad	Complejidad ciclomática
	Facilidad de mantenimiento	Factores de diseño
	Facilidad de entendimiento	Porcentaje de comentarios internos
		Convenciones de nomenclatura variables
		Porcentaje de componentes reutilizados
	Reutilización	Porcentaje reutilizado del componente
	Documentación	Índice de legibilidad
Efectividad del CC	Asignación de recursos	Porcentaje en horas de personal por actividad
	Tasa de terminación	Tiempo de terminación actual *vs.* presupuestado
	Efectividad de la revisión	Vea las métricas de revisión
	Prueba de efectividad	Número de errores encontrados y su criticidad
		Esfuerzo requerido para corregir un error
		Origen del error

Aseguramiento estadístico de la calidad del software

- Implica los siguientes pasos:
  - Se recolecta y clasifica información sobre los errores y defectos de software.
  - Se hace un intento por rastrear cada error y defecto hasta su causa subyacente (por ejemplo, que no se conforma a las especificaciones, error en el diseño, violación de estándares, comunicación deficiente con el cliente).
  - Mediante el uso del principio de Pareto (el 80% de los defectos puede rastrearse hasta el 20% de todas las posibles causas), aislar el 20% (las poco esenciales).
  - Una vez identificados las causas esenciales, pasar a corregir los problemas que provocaron los errores y defectos.

Aseguramiento estadístico de la calidad del software

- Causas:
  - Especificaciones incompletas o erróneas. (EIE)
  - Mala interpretación de la comunicación con el cliente (MCC).
  - Desviación intencional de las especificaciones (DIE).
  - Violación de los estándares de programación (VES).
  - Error en la representación de los datos (ERD).
  - Interfaz de componentes inconsistente (ICI).
  - Error en la lógica de diseño (ELD).
  - Prueba incompleta o errónea (PIE).
  - Documentación imprecisa o incompleta (DII).
  - Error en la traducción a lenguaje de programación del diseño (TLP).
  - Interfaz humano/computadora(IHC) ambigua o inconsistente.
  - Misceláneos (MIS).

Seis Sigma para la ingeniería de software

- Seis Sigma es la estrategia que más se utiliza en el aseguramiento estadístico de la calidad en la industria, define tres pasos básicos:
  - Definir los requerimientos del cliente, los entregables y las metas del proyecto mediante métodos bien definidos de comunicación con el cliente.
  - Medir el proceso existente y su resultado para determinar el desempeño actual de la calidad.
  - Analizar las métricas de defectos y determinar las causas esenciales.

Confiabilidad del software

- No cabe duda de que la confiabilidad de un programa de computadora es un elemento importante de su calidad en general. Si un programa fracasa de manera repetida y frecuente, importa muy poco si otros factores de la calidad del software son aceptables.
  - Medidas de confiabilidad y disponibilidad: Una simple medida de confiabilidad es el tiempo medio entre falla (MTBF).
    - MTBF=MTTF+MTTR
  - Uso de la IA para modelar la confiabilidad: algunos ingenieros de software ven la ciencia de datos como la aplicación de técnicas de inteligencia artificial para resolver problemas de ingeniería de software. Una de las cosas que intentan realizar los métodos de inteligencia artificial es proveer soluciones razonables a problemas en donde los datos necesarios pueden estar incompletos.
  - Seguridad del software: es una actividad de aseguramiento de la calidad que se enfoca en la identificación y evaluación de riesgos potenciales que pueden afectar al software de manera negativa y provocar la falla de todo un sistema. Si los riesgos pueden identificarse en las primeras etapas del proceso, pueden especificarse características de diseño de software que eliminen o controlen los riesgos potenciales.
  - Los estándares de calidad ISO 9000: un sistema de aseguramiento de la calidad puede definirse con la estructura organizaciones, las responsabilidades, los procedimientos, procesos y recursos para implementar la gestión de la calidad [ANS87]. Los sistemas de aseguramiento de la calidad se crean para ayudar a las organizaciones a asegurar sus productos y servicios a satisfacer las expectativas del cliente cumpliendo con sus especificaciones. Estos sistemas cubren una amplia variedad de actividades que abarcan toda la vida útil de un producto, incluida la planeación, el control, la medición, la prueba y la generación de informes.

Confiabilidad del software

INFORMACIÓN

El estándar ISO 9001:2015

A continuación se definen los elementos básicos del estándar ISO 9001:2015. Puede obtener información detallada sobre el estándar de la Organización Internacional de Normalización (**www.iso.ch**) y de otras fuentes de internet (como **www.praxiom.com**).

Establecer los elementos de un sistema de gestión de calidad.

- Desarrollar, implementar y mejorar el sistema.
- Definir una política que haga énfasis en la importancia del sistema.

Documentar el sistema de calidad.

- Describir el proceso.
- Producir un manual operacional.
- Desarrollar métodos para controlar (actualizar) los documentos.
- Establecer métodos para mantenimiento de registros.

Dar soporte al control y aseguramiento de calidad.

- Promover la importancia de la calidad entre todas las partes interesadas.
- Enfocarse en la satisfacción del cliente.
- Definir un plan de calidad que aborde objetivos, responsabilidades y autoridad.
- Definir mecanismos de comunicación entre las partes interesadas.

Establecer mecanismos de revisión para el sistema de gestión de calidad.

- Identificar métodos de revisión y mecanismos de retroalimentación.
- Definir procedimientos de seguimiento.

Identificar los recursos de calidad, incluido el personal, la capacitación y elementos de la infraestructura.

Establecer mecanismos de control.

- Para planeación.
- Para los requerimientos del cliente.
- Para actividades técnicas (como análisis, diseño, pruebas).
- Para monitoreo y gestión de proyectos.

Definir métodos para reparación.

- Evaluar datos y métricas de calidad.
- Definir el enfoque para un proceso continuo y mejora de la calidad.

Cuestionario

- Describa cómo evaluaría la calidad de una universidad antes de hacer solicitud para entrar en ella. ¿Qué factores serían importantes?¿Cuáles serían imprescindibles?
- Con base en la definición del proceso de calidad de software ¿cree usted que sea posible generar un producto útil que proporcione un valor cuantificable sin usar un proceso efectivo?
- Describa el dilema de la calidad de software con sus propias palabras.
- ¿Qué es el software “suficientemente bueno”? Nombre una empresa en concreto y especifique productos que crea que se hayan desarrollado mediante la filosofía de “suficientemente bueno”.
- Realice una búsqueda en web y encuentre otros tres ejemplos de “riesgos” para el público que puedan rastrearse de manera directa al software de mala calidad. Considere iniciar su búsqueda en http://catless.ncl.ac.uk/risks.
- ¿La calidad y la seguridad son lo mismo? Explique.

re

Cuestionario

- Explique la diferencia entre un error y un defecto.
- ¿Por qué no podemos solo esperar hasta la prueba para encontrar y corregir todos los errores de software?
- Suponga que se introdujeron 10 errores en el modelo de requerimientos y que cada error se amplificara por un factor de 2:1 en el diseño, que se introducen 20 errores de diseño adicionales y luego se amplifican por 1.5:1 en el código, en donde se introducen 30 errores adicionales. Suponga además que todas las pruebas unitarias encontrarán el 30% de todos los errores, que la integración encontrará el 30% de los errores restantes y que las pruebas de validación encontrarán el 50% de los errores que faltan. No se realizan revisiones. ¿Cuántos errores se liberarán al campo?
- En el caso anterior, si cuesta $4800 encontrar y corregir cada uno de los errores que se liberan al campo, y cuesta $240 encontrar y corregir cada error detectado en la revisión. ¿Cuánto dinero nos ahorramos si realizamos las revisiones?

Cuestionario

- ¿Puede pensar en algunas instancias en las que un control documental pudiera crear problemas en vez de proveer beneficios?
- Algunas personas dicen que “el control de la variación es el corazón del control de calidad”. Puesto que cada programa creado es distinto de cualquier otro, ¿cuáles son las variaciones que buscamos y cómo las controlamos?
- ¿Es posible evaluar la calidad del software si el cliente sigue cambiando lo que se supone que debe hacer?
- Calidad y confiabilidad son conceptos relacionados, pero son fundamentalmente distintos en varias formas. Comente sobre esas diferencias.

re

Cuestionario

- Puede un programa ser correcto y de todas formas no ser confiable. Explique
- Puede ser un programa ser correcto y de todas formas no exhibir una buena calidad?. Explique.
- ¿Por qué hay tensión con frecuencia entre un grupo de ingeniería de software y un grupo independiente de aseguramiento de la calidad del software? ¿es esto saludable?

re

Referencias

- 9na edición Ingeniería de software un enfoque práctico Roger S. Pressman - Bruce R. Maxim, Mc Graw Hill.

GRACIAS