Modelos de Calidad de Software

Tema Nº3:Introducción a Calidad de Software – Parte 3

Indicador de logro Nº3:Diferencia la calidad y pruebas y software, considerando el ciclo de vida de las pruebas y sus principios.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3:**

Introducción a Calidad de Software – Parte 3

**Subtema 3.1:**

Pruebas de software

Las pruebas de software (Software Testing) comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.



¿Con cuantas pruebas es suficiente?

* No existe un número de pruebas fijo
* Depende de muchos factores:
  + Tipo de software
  + Experiencia de los probadores
  + Tiempo
  + Dinero

Causas de los defectos de software:

* Errores humanos
* Errores técnicos
* Errores ambientales o de entorno

Error àDefecto à Puede provocar o no fallo

Si provoca fallo à PROBLEMA

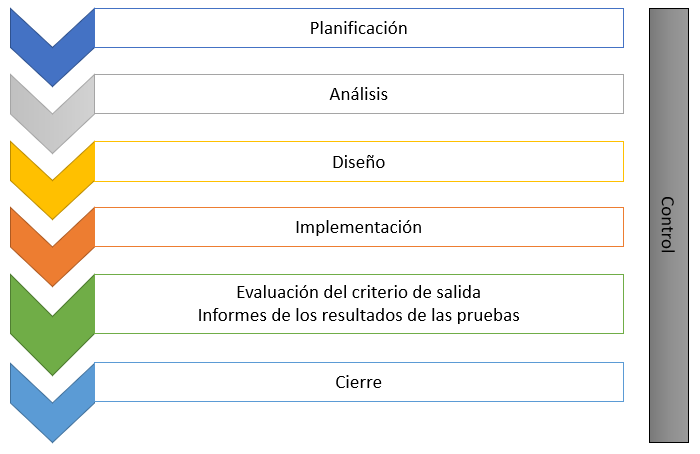
**Subtema 3.2:**

Ciclo de vida de las pruebas de software

Un ciclo de vida es la secuencia de cambios que atraviesa una entidad de una forma a otra. Estas actividades incluyen verificar el software desarrollado para ver si cumple con los requisitos específicos. Si hay algún defecto en el producto, los probadores trabajan con el equipo de desarrollo.

El programa de estudios del ISQB nivel básico propone un framework (marco de trabajo) genérico del proceso de pruebas, Éste no es prescriptivo porque no impone un esquema de madurez y reconoce el potencial de omisión de algunos elementos del framework en algunas circunstancias.

El proceso de pruebas del ISTQB incluye los siguientes pasos:



Dado que el proceso de pruebas tiene que ser adaptado al ciclo de vida del software, el proceso permite la superposición, la concurrencia e incluso la iteración en estos pasos.

Etapas del ciclo de vida de las pruebas:

**La planificación incluye las siguientes actividades:**

* Determinar el alcance de las pruebas, los riesgos, los objetivos y las estrategias.
* Determinar los recursos de las pruebas necesarios.
* Implementar las estrategias de las pruebas.
* Crear un cronograma del análisis y el diseño de las pruebas.
* Crear un cronograma de la implementación, la ejecución y la evaluación de las pruebas.
* Determinar los criterios de salida de las pruebas.

**El control incluye las siguientes actividades:**

* Medir y analizar los resultados.
* Monitorear y documentar el progreso, la cobertura y los criterios de salida de las pruebas.
* Iniciar acciones correctivas.
* Tomar decisiones.

Tenga en cuenta que muchas actividades de planificación y control implican la obtención del acuerdo, el soporte y el consenso del equipo del proyecto y la gerencia del proyecto.

Pasemos ahora al análisis y el diseño de pruebas, que consisten principalmente en actividades de contribuyente individual.

**El análisis incluye las siguientes actividades:**

* Revisar la base de pruebas. La base de pruebas es aquella en la cual las pruebas se basan, con frecuencia, incluyendo los requisitos o las especificaciones de diseño, las arquitecturas de red o sistema o los riesgos de calidad.
* Identificar y priorizar las condiciones de pruebas, los requisitos de pruebas o los objetivos de pruebas y los datos de prueba necesarios. Esto requerirá a menudo un análisis de los ítems de pruebas, así como su comportamiento, especificación y estructura. Por supuesto, estos ítems podrían no existir todavía, entonces usted podría estar dependiendo de sus descripciones, así como los casos de uso.
* Evaluar la comprobabilidad de los requisitos y el sistema. Esto podría resultar en la creación de informes acerca de algunos asuntos acerca de estas áreas para la gerencia.

**El diseño incluye las siguientes actividades:**

* Diseñar y priorizar combinaciones específicas de datos de prueba, acciones y resultados esperados para cubrir la base de pruebas, los riesgos importantes de calidad, y cualquier otra cosa que necesite cobertura.
* Identificar los datos de prueba necesarios para las condiciones establecidas.
* Diseñar los casos de prueba.
* Diseñar el entorno de pruebas.
* Identificar alguna infraestructura y algunas herramientas necesarias.

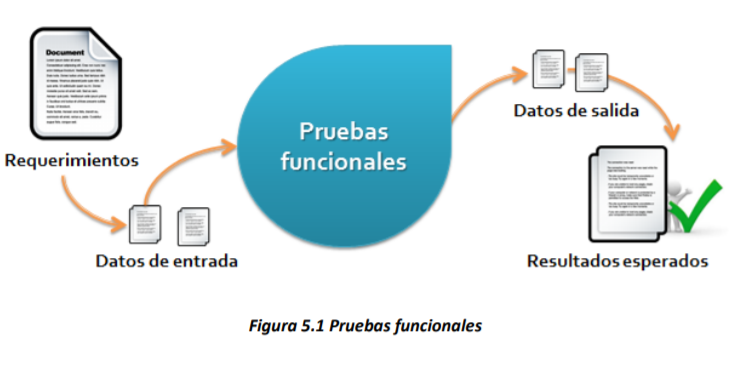
Continuemos con la implementación y ejecución de las pruebas, las cuales, además consisten principalmente de las actividades de un contribuyente individual.

**La implementación incluye las siguientes actividades:**

* Desarrollar, implementar y priorizar casos de prueba, crear datos de prueba y escribir procedimientos de prueba.
* Preparar arneses de prueba y escribir scripts de pruebas automatizadas.
* Organizar juegos de prueba y secuencias de procedimientos de prueba para la ejecución eficiente de las pruebas, teniendo en cuenta las diversas restricciones que podrían determinar el orden en el cual las pruebas deben ser ejecutadas.
* Verificar que el entorno de pruebas ha sido instalado correctamente.

**La ejecución incluye las siguientes actividades:**

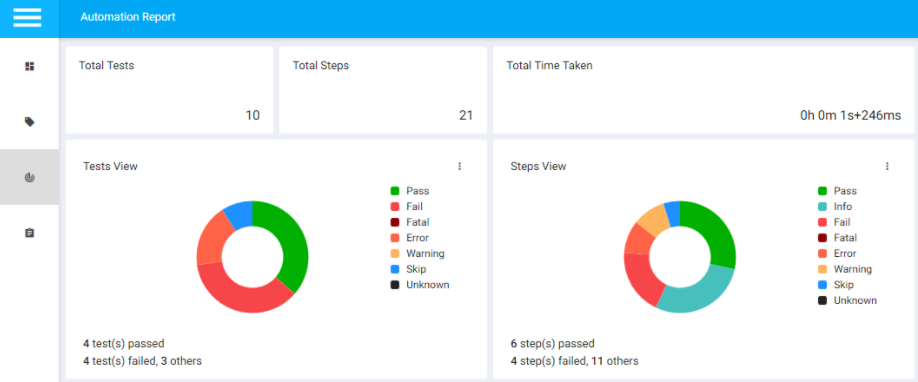
* Ejecutar casos de prueba tanto manuales como automatizados.
* Registrar los resultados de las pruebas, incluyendo las versiones del software sometido a pruebas, las herramientas de pruebas y el testware.
* Comparar los resultados reales y esperados, lo cual podría requerir la identificación de anomalías donde los resultados reales y esperados no coinciden.
* La investigación de anomalías puede resultar en la creación de informes y el análisis de las incidencias.
* Repetir las pruebas corregidas y/o actualizadas donde sea necesario.
* Ejecutar las pruebas de confirmación y/o regresión, cuando las nuevas versiones de pruebas lleguen.



**La evaluación de los criterios de salida y la creación de informes de los resultados de las pruebas** están fuertemente superpuestas con la ejecución de las pruebas, las cuales consisten principalmente en actividades de gestión de actividades de gestión, incluyendo las siguientes:

* Comprobar los registros de las pruebas contra los criterios de salida de las pruebas especificados durante la planificación de pruebas.
* Evaluar si son necesarias más pruebas o si los criterios de salida especificados deben ser modificados.
* Escribir un informe del resumen de las pruebas para los interesados del negocio.

A medida que la ejecución de las pruebas llega a un cierre, los criterios de salida han sido cumplidos y los informes finales de los resultados de las pruebas son recopilados, las actividades del cierre comienzan a ocurrir.



**Cierre.**

Consiste en actividades de gestión y actividades de contribuyente individual, incluyendo:

* Confirmar los entregables de pruebas, la resolución final o postergación de los informes de defectos y la aceptación del sistema por las partes que lo reciben.
* Finalizar y archivar el testware, el entorno de pruebas y la infraestructura de pruebas para su uso posterior durante el mantenimiento.
* Entregar el testware y la posibilidad de ítems adicionales.
* Realizar un estudio retrospectivo para tomar en cuenta las mejoras para futuras versiones, los proyectos y los procesos de pruebas.

**Ejemplos:**

* Validar que se efectue una transacción de manera correcta y efectue la contabilidad.
* Verificar que el mensaje de bienvenida del sistema diga “Hola mundo”.
* Que pasaria si ingreso una cuenta bancaria incorrecta? El sistema deberia arrojar un error.
* Que sucede si al vez de presionar guardar,presiono el boton cancelar?

**Subtema 3.3:**

Los 7 principios de las pruebas de software



**1. Las pruebas demuestran la presencia de defectos.**

¿Es posible demostrar, con total certeza, que NO hay defectos en un software?

La respuesta es un NO rotundo. Las pruebas pueden demostrar la existencia de incidencias en un producto, que una vez solventadas, reducen la probabilidad de que defectos no descubiertos permanezcan en nuestro sistema.

¡Cuántas veces nos habremos asombrado con los bugs más inverosímiles que los usuarios finales del software que hemos probado son capaces de encontrar!

* El proceso de pruebas puede probar la presencia de defectos.
* Las desviaciones identificadas a lo largo del proceso de pruebas demuestran la presencia de un fallo.
* El proceso de pruebas no puede demostrar la ausencia de defectos.
* Las pruebas reducen la probabilidad de la presencia de defectos que permanecen sin ser detectados.
* La ausencia de fallos no demuestra que un producto software no tenga defectos.
* El mismo proceso de pruebas puede contener errores.
* Las condiciones de las pruebas pueden ser inapropiadas para detectar errores.

Lo que podemos sacar de este principio es que, por más que probemos una aplicación nunca podemos decir que el sistema se encuentra al 100% en su calidad de software, y esto lo decimos es porque no podemos estar seguros de que la aplicación ya no tiene defectos.

**2. Las pruebas exhaustivas no existe o son imposibles**

No es posible probar todas las funcionalidades con todas las combinaciones válidas e inválidas de datos de entrada durante la prueba real. En lugar de este enfoque, se considera que la prueba de algunas combinaciones se basa en la prioridad utilizando diferentes técnicas.

Las pruebas exhaustivas requerirán esfuerzos ilimitados y la mayoría de esos esfuerzos son ineficaces. Además, los cronogramas del proyecto no permitirán probar tantas combinaciones. Por lo tanto, se recomienda probar los datos de entrada utilizando diferentes métodos como el Particionamiento de Equivalencia y el Análisis de Valor Límite.

Por ejemplo, Supongamos que tenemos un campo de entrada que acepta alfabetos, caracteres especiales y números del 0 al 1000 únicamente. Imagínese cuántas combinaciones aparecerían para probar, no es posible probar todas las combinaciones para cada tipo de entrada.

Los esfuerzos de prueba necesarios para probar serán enormes y también afectarán el cronograma y el costo del proyecto. Por eso siempre se dice que prácticamente no es posible realizar pruebas exhaustivas.

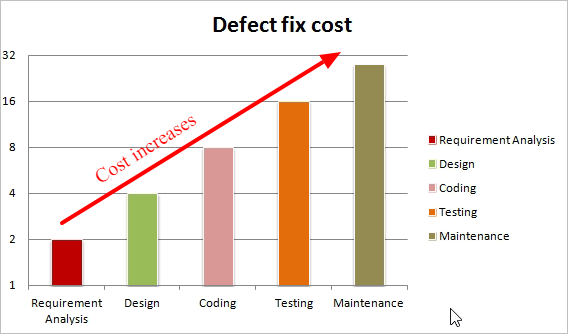


**3. Pruebas tempranas**

Los probadores deben involucrarse en una etapa temprana del ciclo de vida del desarrollo de software (SDLC). De esta forma se pueden identificar los defectos durante la fase de análisis de requisitos o cualquier defecto de documentación. El costo involucrado en la reparación de tales defectos es muy inferior en comparación con los que se encuentran durante las últimas etapas de la prueba.

* La corrección de un defecto es menos costosa en la medida en la cual su detección se realiza en fases más tempranas del proceso software.
* Se obtiene una máxima rentabilidad cuando los errores son corregidos antes de la implementación.
* Los conceptos y especificaciones pueden ser probados.
* Los defectos detectados en la fase de concepción son corregidos con menor esfuerzo y costos.
* La preparación de una prueba también consume tiempo.
* El proceso de pruebas implica más que sólo la ejecución de pruebas.
* Las actividades de pruebas pueden ser preparadas antes de que el desarrollo se haya completado.
* Las actividades de pruebas (incluidas las revisiones) deben ser ejecutadas en paralelo o la especificación y diseño software.

Considere la siguiente imagen que muestra cómo aumenta el costo de la reparación de defectos a medida que las pruebas avanzan hacia la producción en vivo.



**4. Agrupación de defectos**

Durante las pruebas, puede suceder que la mayoría de los defectos encontrados estén relacionados con una pequeña cantidad de módulos. Puede haber varias razones para esto, como que los módulos pueden ser complejos, la codificación relacionada con dichos módulos puede ser complicada, etc.

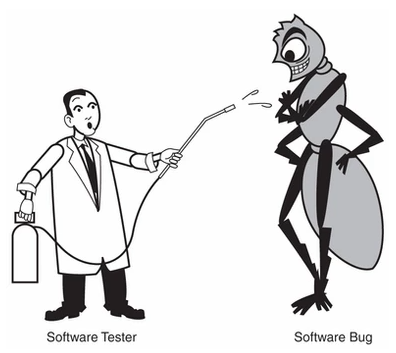
Este es el Principio de Pareto de las pruebas de software donde el 80% de los problemas se encuentran en el 20% de los módulos. Aprenderemos más sobre la agrupación de defectos y el principio de Pareto más adelante en este artículo.

* ¡Encuentre un defecto y encontrará más defectos "cerca"!
* Los defectos aparecen agrupados como hongos o cucarachas.
* Cuando se detecta un defecto es conveniente investigar el mismo módulo en el que ha sido detectado.
* Los probadores ("testers") deben ser flexibles.
* Habiendo sido detectado un defecto es conveniente volver a considerar el rumbo de las pruebas siguientes. La identificación de un defecto puede ser investigada con un mayor grado de detalle, por ejemplo, realizando pruebas adicionales o modificando pruebas existentes.

**5. Paradoja del pesticida**

El principio de la paradoja de plaguicidas dice que, si el mismo conjunto de casos de prueba se ejecuta una y otra vez durante el período de tiempo, entonces este conjunto de pruebas no son lo suficientemente capaces de identificar nuevos defectos en el sistema.

Para superar esta “paradoja de los plaguicidas”, el conjunto de casos de prueba debe revisarse y revisarse periódicamente. Si es necesario, se puede agregar un nuevo conjunto de casos de prueba y los casos de prueba existentes se pueden eliminar si no pueden encontrar más defectos en el sistema.



**6. Las pruebas dependen del contexto**

Hay varios dominios disponibles en el mercado como banca, seguros, médicos, viajes, publicidad, etc. y cada dominio tiene una serie de aplicaciones. Además, para cada dominio, sus aplicaciones tienen diferentes requisitos, funciones, diferentes propósitos de prueba, riesgo, técnicas, etc.

Los diferentes dominios se prueban de manera diferente, por lo que la prueba se basa exclusivamente en el contexto del dominio o la aplicación.

Por ejemplo, probar una aplicación bancaria es diferente a probar cualquier aplicación de publicidad o comercio electrónico. El riesgo asociado con cada tipo de aplicación es diferente, por lo que no es efectivo utilizar el mismo método, técnica y tipo de prueba para probar todos los tipos de aplicación.

**7. Falacia de ausencia de errores**

Si el software se prueba completamente y si no se encuentran defectos antes del lanzamiento, podemos decir que el software está libre de defectos en un 99%. Pero, ¿qué pasa si este software se prueba con requisitos incorrectos? En tales casos, incluso encontrar defectos y arreglarlos a tiempo no ayudaría ya que las pruebas se realizan en requisitos incorrectos que no se ajustan a las necesidades del usuario final.

Por ejemplo, Supongamos que la aplicación está relacionada con un sitio de comercio electrónico y los requisitos de la funcionalidad 'Carrito de compras o Carrito de compras' se interpretan y prueban incorrectamente. Aquí, incluso encontrar más defectos no ayuda a mover la aplicación a la siguiente fase o al entorno de producción.

**Actividad:**

Los estudiantes desarrollarán ejercicios\* para profundizar los conceptos realizados en la guía 3:

1. ¿Qué son las pruebas de software?

2. ¿Cuál es la importancia de las pruebas?

3. Con sus propias palabras explique los 7 principios de pruebas.

4. ¿Cuáles son los entregables por capa etapa de las pruebas?