



INSTITUTO TECNOLÓGICO DE PABELLÓN DE ARTEAGA

Pruebas de Software

“Resumen PDF 1”



ALUMNA:

NAYELI JAZMIN CONTRERAS HERNÁNDEZ

MAESTRO:

ANA CRISTINA TELLO VAZQUEZ

15–NOVIEMBRE -2022

Pruebas De Software

Tarea 1^o Unidad 3^o

15 / Nov. / 22

Nayeli Joaquin Contreras Hernández

Ciclo de vida de las pruebas de software.

Las pruebas de software se refiere a un proceso de prueba con pasos específicos, se deben realizar los pasos en un orden específico para garantizar que el software cumpla con los objetivos de calidad. Las fases de STLC son diferentes para cada organización.

Las fases son las siguientes:

- + Analisis de requisitos
- + Fase de planificación
- + Integración de casos de prueba
- + Configurar entorno de prueba.
- + Fase de implementación
- + Cierre del ciclo de prueba.

Analisis de requisitos

- El equipo de control de calidad se comunica con diferentes partes interesadas (cliente, analista de negocios, líderes, técnicos, arquitectos del sistema) para comprender los requerimientos. Generando tablas con los detalles de cada requerimiento así como si son funcionales o no.

Planificación de Pruebas.

- Un gerente de control de calidad generalmente determina los esfuerzos y las estimaciones de costos para el proyecto y completar el plan de prueba, también las estrategias de prueba.

Complendo con actividades y entregables.

Actividades:

- * Preparación del plan de prueba.
- * Selección de herramientas.
- * Estimación de los esfuerzos.
- * Planificación de recursos y determinación de roles.

Entregables:

- * Plan de prueba
- * Estimación del esfuerzo

Casos de Prueba.

- Se crean, verifican y reelaboran los casos de prueba y los scripts de prueba.

Actividades

- * Crear casos de prueba
- * Revisión y referencia de casos de prueba
- * Crear datos de prueba

Entregables

- * Casos de prueba / scripts
- * Datos de prueba.

Entorno de prueba.

- La configuración del entorno de prueba es uno de los aspectos críticos del proceso de prueba, el equipo de prueba se deberá usar con una prueba de preparación (prueba de humo) del entorno de prueba para determinar si es satisfactorio.

Actividades

- * Comprende la arquitectura (tiene el entorno necesario, realizando una lista de los requisitos de hardware y software).
- * Prueba de configuración del entorno
- * Realice prueba de humo.

Entregables

- * Configuración del entorno y datos de prueba listos
- * Resultados de prueba de humo.

Ejecución de Prueba.

- los evaluadores realizan las pruebas según los planes de prueba y los casos de prueba preparados se reportan errores de desarrollo al corregir se realizan las pruebas nuevamente.

Actividades

- * Realizar pruebas de acuerdo al plan
- * Documentar resultados
- * Probar las correcciones
- * Seguir los defectos hasta terminar

Entregables

- * Casos de prueba actualizado con resultados
- * Reporte defectos.

Cierre de pruebas

- El equipo de prueba analiza y discute las estrategias, se planea reducir los errores para proyectos similares.

Actividades

- * Evaluar los criterios en función del tiempo.
- * Preparar estrategias basadas en parámetros.
- * Documentar detalles del proyecto.
- * Preparar el informe.

Entregables

- * Informe final de prueba.

Estrategias de las pruebas de software.

- Es un conjunto de pasos que incluyen métodos de prueba y técnicas de pruebas estratégicas.

Características generales:

- ✓ las pruebas comienzan en los componentes.
- ✓ Prueba y depuración son diferentes pero la depuración debe incluirse en cualquier estrategia de prueba.
- ✓ Pruebas validadas donde los requerimientos fueron establecidos como parte de su modelo.
- ✓ Se lleva a prueba el sistema de forma que el software y otros elementos se prueben como un todo.

Plan de Pruebas

- Se elabora para atender los objetivos de calidad en un desarrollo de sistemas, encargándose de definir aspectos como tipos de prueba, criterios de aceptación, infraestructura, y planificación.

Analizar los requerimientos

- Con la elaboración de un plan de pruebas los requerimientos de usuario que componen el proyecto verifican la calidad.

Se deberá analizar la información incluyendo la matriz de trazabilidad, especificaciones y diseño funcional, historias de usuarios, entre otra documentación.

Identificar las funcionalidades nuevas.

- la documentación del análisis de requisitos y de las entrevistas deben identificar en el plan de pruebas de software la lista de funcionalidad

Identificar las funcionalidades de sistemas existentes.

- Como identificar las situaciones que pueden existir:
 - funcionalidades modificadas de cada usuario, cambio de pantallas o cambio de flujo de procesos que se incluyen en el plan de pruebas.
 - funcionalidades modificadas en sus componentes: se modifican los componentes internos que comparten con otras funcionalidades del sistema

Tipos de Pruebas

1 Pruebas funcionales: un sistema que se describen en las especificaciones de requisitos.

- Completitud funcional
- Corrección funcional
- Pertenencia funcional

2 Pruebas no funcionales: Tienen el comportamiento externo del software: es como funciona el sistema y se suele usar técnicas de diseño.

- Pruebas de carga
- Pruebas de rendimiento
- de volumen
- de esfuerzo
- de seguridad
- de estabilidad
- de compatibilidad
- de usabilidad.

3 Pruebas de caja blanca: Es una técnica de diseño que usa la estructura de control para obtener los casos de prueba.

Aportaciones:

- Revisar condiciones lógicas
- Revisar la estructura de datos.

4 Pruebas unitarias: Es la verificación de unidades de software de forma aislada, es decir, probar el correcto funcionamiento de una unidad de código.

5 Pruebas de integración: se ocupan de probar las interfaces entre los componentes, las interacciones con distintas partes de un mismo sistema.

6 Pruebas de validación o adaptación: son responsabilidad del cliente y es la única parte en la que está involucrada. Existen 4 formas de aceptación:

- Aceptación del contrato
- Aceptación de usuario
- Pruebas operativas
- Pruebas alfa y beta.

Herramientas de Pruebas: Existen herramientas para diseñar casos de pruebas, gestionar y administrar pruebas y monitorizar sistemas de prueba.

- Categorías de herramientas

- Pruebas estáticas
- Planificación y gestión de pruebas
- Pruebas de automatización
- Drivers y stubs
- Pruebas de carga y rendimiento
- Monitorización y seguridad.

Etapas tempranas del proyecto.

- PDM (libre): analizador de código fuente que encuentra defectos comunes de programación
- Checkstyle: Herramienta de chequeo de código Java que cumple con estándares de codificación
- Simion: Detector de software duplicados.



INSTITUTO TECNOLÓGICO DE PABELLÓN DE ARTEAGA

Pruebas de Software

“Resumen PDF 2”



ALUMNA:

NAYELI JAZMIN CONTRERAS HERNÁNDEZ

MAESTRO:

ANA CRISTINA TELLO VAZQUEZ

15–NOVIEMBRE -2022

Estrategia y técnicas de prueba del software...

Tarea 1ª Unidad: 3º
15 / Nov / 22
Naueli Jaemin Contreras Abz

Las pruebas de software son una parte importante pero considerada costosa del proceso de desarrollo de software.

Existen dos objetivos principales del proceso

- * Maximizar el número de errores detectados (cobertura).
- * Reducir al mínimo de casos de prueba (costo).

Pruebas unitarias:

- Emplean técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes.

Para probar cada módulo se requiere de un módulo controlador y de uno o más módulos simulados (stubs). Una herramienta libre para escribir pruebas unitarias en Java es www.junit.org que soporta la ejecución de pruebas en modo batch o mediante GUI, sin embargo para probar una clase se escribe una clase de prueba y a la clase de pruebas unitario está compuesto de un método de prueba que se puede identificar con un @test.

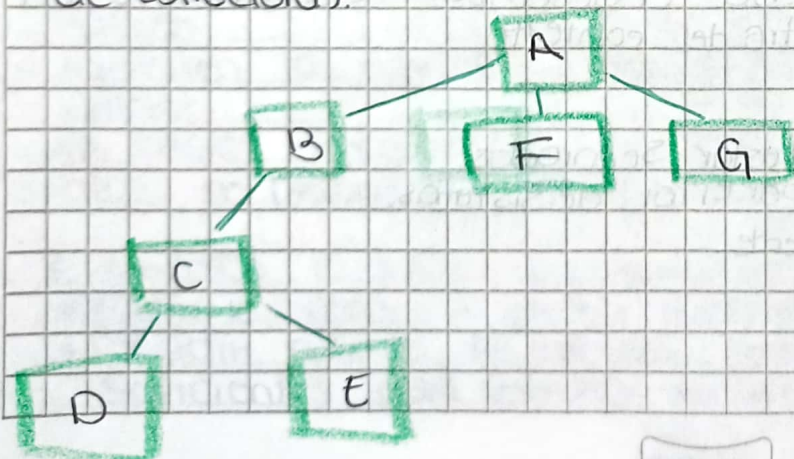
Pruebas de integración:

- Se utilizan técnicas que verifican el correcto manejo de las entradas y salidas del software.

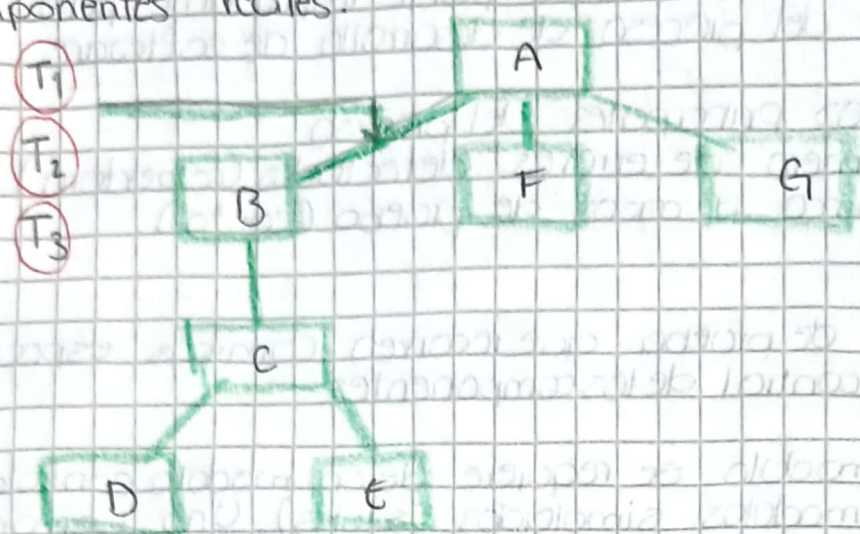
Objetivos principales:

- Descubrir errores asociados con la interfaz de los módulos.
- Ensamblar los módulos individuales para formar sub-sistemas y al final un sistema completo.

El enfoque utilizado es el de grafo de llamadas. (Diagrama de estructura).



Existen módulos simulados que se reemplazan uno a la vez con los componentes reales.



La construcción y pruebas con los módulos en los niveles más bajos de la estructura del programa, no se requiere el uso de módulos simulados.

Los módulos a niveles bajos de la jerarquía son combinados en grupo que realizan una subfunción específica del software. los controladores son reemplazados y los grupos son cambiados hacia arriba en la estructura del programa.

- Pruebas de validación: se enfoca en los requerimientos
- Pruebas de recuperación: fuerza el software a fallar de diferentes formas.
- Pruebas de seguridad: verifica que los mecanismos de protección integrados en el sistema realmente.
- Pruebas de resistencia: ejecutan un sistema de manera que se demanden recursos en cantidades enormes.
- Prueba de desempeño: prueba el desempeño del software en tiempo de ejecución dentro del contexto

Depuración

- Síntomas y causas pueden estar separados.
- Causa puede darse por error de sistemas.
- Sistema puede estar intermitente.

Existen diferentes herramientas de depuración, en la mayoría de las IDEs se incluye alguna como Java y NetBeans.

Valgrind: es una herramienta que permite detectar fácilmente errores (bugs) en el manejo de hilos (threads). Funciona con programas escritos en cualquier lenguaje o mechas de ellos que trabaja directamente con el código binario.

Proyectos importantes:

- Open Office
- Mozilla Firefox
- MySQL
- PHP
- Perl

La valgrind es muy útil para detectar los tipos de errores en el uso de memoria.

- Escritorio en memoria no inicializada.
- Lectura de memoria fuera de rango.
- Condicionantes que dependen de memoria.
- Incorrecta liberación de memoria.

Métodos de prueba:

- De caja blanca (o estructurales)
- De caja negra (o funcionales)
- Caja blanca: Verifica correcta implementación de las unidades internas las estructuras y sus relaciones.
- Caja negra: Verifican el correcto manejo de funciones externas provistos o soportan por es software.

Caja blanca:

- Ejecuten los lógos verdaderos y falsos de todas las decisiones lógicas.
- Ejerciten las estructuras de datos internas para asegurar su validez.

Los errores tipográficos son aleatorios, es probable que los datos no probados contengan algunos errores de este tipo.

Elementos a enfocar las pruebas.

- Elegir los datos de prueba adecuados.
- Decidir cuando los esfuerzos dedicados a las pruebas son suficientes para terminar el proceso.

Criterios de suficiencia de las pruebas:

- los basados en las propiedades estructurales de los programas (estructuras lógicas y de control, flujo de los datos).
- los basados en las especificaciones de los programas.

Prueba de la estructura de control:

- Prueba de condición, un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en un modelo.
- Prueba de flujo de datos, selecciona rutas de prueba del módulo de acuerdo con la localización de las definiciones y utilización de las variaciones del programa.

Condiciones mínimas

- Iniciar en el ciclo más interno. Asignar a todos los ciclos sus valores de iteración mínimos.
- Es un método de diseño de casos de prueba que se enfoca en el rol de las variables (datos) en el código.
- Selecciona pruebas del módulo de acuerdo con la localización de las definiciones y utilización de las variables.

Ejemplo:

1º Cuando el código se hace más complejo y hay más variables que considerar se vuelve mucho más tardado analizar sus roles en el flujo de datos, identificar rutas y diseñar casos de prueba.

2º Pruebas basadas en flujo de datos ocurre en el caso de variables dinámicas como los punteros.

3º No existen herramientas comerciales que provean soporte a este tipo de pruebas.

Caja negra:

Permite derivar casos de prueba que buscan encontrar los siguientes tipos de errores:

- funciones incorrectas
- Errores en estructuras de datos o en acceso a BD externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización o terminado

Cardo modulos de software tiene un dominio de entrada del cual los datos de prueba deben ser seleccionados, existen reportados en la literatura diversas formas de seleccionarlos. Si dichos datos de prueba se seleccionan de forma aleatoria entonces hablamos de pruebas aleatorias.

El uso de pruebas aleatorias puede ahorrar mucho tiempo y esfuerzo en la seleccion de los datos.

Ventajas:

- Elimina las necesidades de las pruebas exhaustivas.
- Es una guía para seleccionar el conjunto de datos de entrada para las pruebas, con alta probabilidad de detectar defectos.
- Permite realizar una cobertura de un dominio de entradas de grandes con un subconjunto pequeño tomado de cada clase de equivalencia.

Consideraciones:

- * Clases de equivalencia tanto validas como invalidas.
- * las clases deben tambien seleccionarse a partir de las condiciones de salida.
- * la derivación de las clases de equivalencia es un proceso heurístico.
- * En algunos casos es difícil identificar las clases de equivalencia debido a una mala especificación del modulo.

El analisis de valores de frontera requiere como su nombre lo dice que se seleccionen valores, permite que los valores tanto en el limite inferior como superior de una clase de equivalencia sea cubierta por los casos de prueba.

las pruebas por partición equivalente la habilidad para desarrollar casos de prueba efectivos requiere de experiencia, son metodos de seleccion de casos de prueba que identifican parametros de entrada individuales y los combinan basados en alguna estrategia.

Los arreglos ortogonales son objetos combinatorios que pueden usarse para representar esos casos de prueba de interacción. Principalmente permiten maximizar el número de errores detectados reduciendo al mínimo el número de casos de prueba.

Los arreglos ortogonales son muy útiles para representar casos de prueba de interacción, representan una buena alternativa frente a las pruebas exhaustivas.

Por lo que los covering arrays no pueden ser empleados para construir casos de prueba para superar esta limitación se emplean otros objetos combinatorios.

Todo los CA como la MCPA tiene otras aplicaciones prácticas:

- Desarrollo de fármacos
- Compresión de datos
- Estudio de la regulación de expresión génica
- Autenticación, etc.