DP2 2021-2022

Knowledge of WIS' testing

Repositorio:

https://github.com/mpadillatabuenca/Acme-Toolkit.git

Miembros:

- José Manuel Bejarano Pozo (josbezpoz@alum.us.es)
- Mario Espinosa Rodríguez (maresprod5@alum.us.es)
- Andrea Meca Sánchez (andmecsan@alum.us.es)
- Manuel Padilla Tabuenca (maresprod5@alum.us.es)
- Ezequiel Pérez Sosa (ezepersos@alum.us.es)
- Javier Terroba Orozco (javteroro@alum.us.es)

GRUPO G1-E2-05

Versión 1.0.0

28/02/2022

Tabla de contenidos

Historial de versiones

Fecha	Versión	Descripción de los cambios
28/02/2022	1.0.0	Creación y finalización del documento.

Introducción

En este documento, realizado por José Manuel Bejarano Pozo hablaré de forma individual sobre los conocimientos previos a la asignatura de Diseño y Pruebas II en cuanto a la realización de pruebas de sistemas de información.

La estructura de los contenidos será de la forma:

- **Introducción a las pruebas:** Finalidad, objetivo, funcionamiento, importancia y tipos de pruebas.
- **Tipos de error en las pruebas:** Tipos de errores que se suelen dar en las pruebas.
- **Estrategias de pruebas:** Distintas opciones de probar la arquitectura de un sistema software.

Introducción a las pruebas

¿Para qué probamos el software? Para comprobar si funciona correctamente y, principalmente, para detectar errores. Es decir, las pruebas de integración es el proceso de ejecutar un programa con la intención de encontrar errores.

Las pruebas de software no permiten garantizar que un programa contiene errores ya que existen demasiadas combinaciones y no se puede probar todo.

El objetivo de las pruebas es detectar el mayor número posible de errores con la mínima cantidad de tiempo y esfuerzo y aumentar nuestra confianza en que el programa cumple los requisitos.

Proceso general de prueba: un sistema a probar recibe una entrada y genera una salida, la cual se comprueba con un oráculo para saber si es correcta o no. Un oráculo es un mecanismo que nos permite saber si el resultado de una prueba es correcto o no. Nos permite saber el resultado esperado.

Importancia de las pruebas: las pruebas pueden consumir hasta el 50% del coste total del sistema. Esto se debe a la necesidad de desarrollar software de calidad.
Complejidad del software:

✓ Producto intangible: no tenemos elementos para saber si está funcionando bien.

✓ Desarrollo a medida: siguiendo las características que quiere el cliente.

✓ Multitud de dispositivos.

Cuanto más tarde detectemos un error, más caro resultará repararlo.

Tipos de pruebas:

✓ Funcionales: fallos relacionados con la funcionalidad del sistema que no cumplen con los requisitos.

✓ No funcionales: fallos relacionados con aspectos no funcionales como rendimiento, usabilidad, etc.

✓ Pruebas unitarias: diseñadas para probar una parte pequeña y bien delimitada de código. Ej: un método.

✓ Pruebas de integración: diseñadas para probar la interacción entre distintos elementos del sistema.

Primero se realizan pruebas unitarias para asegurarnos que cada componente funciona. Ej: varias clases.

✓ Pruebas de sistema: diseñadas para probar el sistema completo.

✓ Pruebas de aceptación: diseñadas para verificar que el sistema cumple con los requisitos de usuario.

Tipos de error en las pruebas

✓ Errores de programación: la integración revela comportamientos inesperados.

✓ Errores debido a asunciones indebidas: un componente hace asunciones incorrectas sobre el funcionamiento de otro. Ej: distintas unidades de medida.

✓ Errores de sincronización: fallos de sincronización entre componentes. Ej: bloqueos.

- Bloqueo mutuo: un recurso queda parado esperando a otro que le mande respuesta. Si la respuesta no llega el recurso queda bloqueado.
- Bloqueo activo.

✓ Errores debidos a incompatibilidad hardware y/o software.

Estrategias de pruebas

Pruebas no incrementales:

En las pruebas no incrementales se integran todos los componentes y se prueba el sistema como un todo. No es una técnica recomendada porque dificulta el aislamiento de errores.

Pruebas incrementales:

En este tipo de pruebas, el programa es construido y probado en pequeños incrementos. Facilita el aislamiento de errores y las pruebas sistemáticas.

Pruebas incrementales descendentes:

La integración se realiza partiendo del punto de entrada y moviéndonos hacia abajo por la jerarquía de control.

Ventajas: permite verificar los puntos de control o decisión (situados arriba en el árbol) al principio del proceso de prueba. La integración en profundidad permite probar funcionalidades completas lo cual aumenta la confianza.

Desventajas: puede requerir el uso de muchos stubs.

✓ Integración descendente-profundidad: la integración se realiza por ramas. Cada rama suele implementar una funcionalidad específica.

¿Qué ocurre si algún módulo/componente aún no ha sido implementado, pero es necesario para probar la integración de otros componentes? Se crean stubs, componentes de mentira que imitan el comportamiento de componentes aún no implementados (implementan su interfaz).

✓ Integración descendente-anchura: la integración se realiza por niveles moviéndose en horizontal por la estructura de control. También requiere el uso de stubs.

Pruebas incrementales ascendentes:

La integración se realiza por niveles partiendo de módulos atómicos situados en los nodos finales de la jerarquía de control. Los componentes se van agrupando en clústeres con una funcionalidad específica. Suele requerir el uso de drivers. Los drivers son programas que aceptan datos de prueba, los pasa a los componentes probados e imprime resultados. Son necesarios cuando el programa que invocará a la funcionalidad que se está probando aún no ha sido implementado.

Ventajas: los puntos de control se prueban en último lugar y se elimina la necesidad de usar stubs.

Desventajas: es necesario el uso de controladores de prueba (drivers).

Pruebas de sandwich:

Combina integración descendente y ascendente.

Conclusión

Las conclusiones obtenidas gracias a la realización de de este documento son principalmente, el pequeño resumen de las destrezas adquiridas a lo largo de mi carrera, pienso que en cuanto a las pruebas de sistemas de información son bastante complejas de implementar de forma correcta ya que suelen producirse una gran cantidad de errores con mucha facilidad y es bastante tedioso de solucionar dichos errores.

Además de todo esto, me gustaría obtener más información sobre los distintos tipos de pruebas, ya que solo conozco las pruebas unitarias.

Espero que en esta asignatura de Diseño y Pruebas 2 pueda aprender mucho más sobre los distintos tipos de pruebas.

Bibliografía

1. Apuntes de la asignatura de Arquitectura e Integración de Sistemas Software (ev.us.es).