

CLASE 11: PRUEBAS DE SOFTWARE

Unversidad Simón Bolívar.
Prof. Ivette Carolina Martínez

Pruebas: Definición

- “Prueba de Software es la ejecución del código usando combinaciones de entradas, en un determinado estado, para **revelar defectos.**”
- “Prueba de Software [...] es el **diseño e implantación de un software especial**: uno que ejercita otro software con la intención de **hallar defectos.**” *Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools (1999)*

En qué consisten las pruebas?



- Determinar **qué partes** del sistema desea probar
- Definir **valores de entrada** que aporten información significativa
- **Correr** el software con los **valores de entrada**
- **Comparar** los **resultados producidos** con los **esperados**
- (Medir características de ejecución: tiempo, memoria usada, etc).

Las pruebas no son ...



- Pruebas \neq corrección

Cuando se descubre un defecto, corrección es el proceso de eliminar el defecto

- Pruebas \neq prueba formal de programas

Las pruebas formales de correctitud son pruebas matemáticas de la equivalencia entre la especificación y el programa

Terminología

- **Falla** – inhabilidad manifiesta sistema para realizar una función necesaria
 - Evidenciado por:
 - Salida incorrecta
 - Terminación anormal
 - Limitaciones de tiempo o espacio incumplidas
- **Defecto** – código incorrecto o faltante
 - Ejecución puede resultar en una falla
- **Error** – acción humana que produce un defecto
- **Problema** – error, falla, defecto

Tipos de Pruebas: según su alcance



- **Prueba Unitaria**

 - alcance: típicamente un ejecutable pequeño

- **Prueba de Integración**

 - alcance: un sistema o subsistema completo de componentes de software y hardware

 - ▣ Ejercita las interfases entre componentes para demostrar que son operables en conjunto

Tipos de Pruebas: según su alcance

□ Prueba de Sistema

alcance: una aplicación completa integrada

▣ Focalizada en características que están presentes sólo al nivel de todo el sistema

▣ Categorías:

■ Funcional

■ Rendimiento

■ Estrés o carga

Tipos de Pruebas: según su intención

- **Prueba dirigida a defectos**

intención: revelar defectos a través de fallas

- Pruebas unitarias e integración

- **Prueba dirigida a Cumplimiento**

intención: demostrar que está conforme con las capacidades requeridas

- Prueba de sistema

- **Prueba de aceptación**

intención: permitir a un usuario/cliente decidir si acepta un producto de software

Tipos de Pruebas: según su intención



□ Prueba de Regresión

Intención: Volver a probar un programa previamente probado, después de algunas modificaciones para asegurarse que no se hayan introducido o aparecido defectos debido a los cambios realizados

□ Pruebas de Mutación

Intención: Introducir defectos a propósito en el software para determinar la calidad de las pruebas

Componentes de una prueba

1. **Caso de Prueba** – especifica:
 - ▣ El estado y ambiente del programa antes de ejecutar la prueba
 - ▣ Las entradas a la prueba
 - ▣ El resultado esperado
2. **Resultados esperados** – qué debe producir el programa:
 - 📄 Valores devueltos
 - 📄 Mensajes
 - 📄 Excepciones
 - 📄 Estado resultante del programa y el ambiente
3. **Oráculo** – produce los resultados esperados del caso de prueba
 - 📄 Puede decidir si se satisfizo la evaluación

Diseño de Casos de Prueba



- Definir los casos de prueba que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo.
- **Pruebas de caja blanca:** Encontrar casos de prueba “viendo” el código interno
- **Pruebas de caja negra:** Encontrar casos de prueba “viendo” los requerimientos funcionales

Pruebas “Caja Blanca”

- Aseguran que la operación interna del programa se ajusta a las especificaciones y que todos los componentes internos se han probado adecuadamente.
 - ▣ Usa la estructura de control para obtener los casos de prueba.
 - ▣ Intentan garantizar que todos los caminos de ejecución del programa quedan probados.
- Pruebas de estructura de control:
 - ▣ Del camino básico: Diseñar un caso de prueba por cada camino independiente
 - ▣ De condición: Diseñar casos de prueba para que todas las condiciones del programa se evalúen a cierto/falso
 - ▣ De bucles: Diseñar casos de prueba para que se intente ejecutar un bucle $0, 1, \dots, n-1, n$ y $n+1$ veces (siendo n el número máximo)

Pruebas “Caja Negra”

- Se centran en los requisitos funcionales del software.
- Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente los requisitos funcionales del programa.
- No son una alternativa a las pruebas de caja blanca.
 - ▣ Complementan a las pruebas de caja blanca → Mejor diseñar los casos de prueba usando los dos tipos de técnicas

Pruebas Caja Negra

Prueba de los valores límite:

- Los errores suelen situarse en los límites.
 - ▣ Si la entrada se encuentra en el rango $a..b$ entonces hay que probar con los valores $a - 1$, a , $a + 1$, $b - 1$, b y $b + 1$
 - ▣ Si la entrada es un conjunto de valores entonces hay que probar con los valores $\max - 1$, \max , $\max + 1$, $\min - 1$, \min y $\min + 1$

Pruebas Caja Negra



Pruebas de la partición equivalente:

- Método de prueba de caja negra que divide el dominio de entrada de un programa en un conjunto de clases de equivalencia de los datos de los que se pueden derivar casos de prueba (al menos uno por cada clase de equivalencia)

Caja negra vs Caja Blanca

Caja Negra

- ❑ No conoce los detalles internos del programa
- ❑ Objetivo: Probar que tan bien el programa está conforme a los requerimientos (cubre todos los requerimientos)

Caja Blanca

- ❑ Conoce la estructura interna del programa
- ❑ Objetivo: Probar que todos los caminos del código están correctos (cubre todo el código)

Caja negra vs Caja Blanca

Caja Negra

- Sólo conoce la especificación
- Se usa típicamente en pruebas de integración y del sistema
- Puede ser realizada por los usuarios

Caja Blanca

- Requiere análisis del código fuente para diseñar los casos de prueba
- Se usa en pruebas unitarias
- Realizadas por programadores

Flujo de trabajo de las pruebas

- **Actividad: planificar pruebas**

Describir una estrategia de prueba, estimar los requisitos y planificar el esfuerzo de la prueba

- **Actividad: diseñar pruebas**

Identificar casos de prueba y procedimientos de prueba

- Diseñar casos de prueba de integración (para verificar que los componentes interaccionan correctamente)
- Diseñar la prueba del sistema (para verificar que el sistema funciona correctamente como un todo)
- Diseñar los casos de prueba de regresión : Al añadir un nuevo módulo puede haber problemas con módulos que antes iban bien. Las pruebas de regresión son un conjunto de pruebas (ya realizadas antes) que aseguran que los cambios no han dado lugar a cambios colaterales.

Flujo de trabajo de las pruebas

- **Actividad: implementar pruebas**

Automatizar los procedimientos de prueba, creando componentes de prueba, si es posible.

- **Actividad: realizar pruebas de integración**

Realizar las pruebas, comparar con los resultados esperados e informar de los defectos

- **Actividad: realizar prueba de sistema**

Se comienzan después de las de integración y se realizan de manera análoga (realizar, comparar e informar)

- **Actividad: evaluar pruebas**

Se comparan los resultados de las pruebas con los objetivos esbozados en el plan de prueba. Hay que preparar métricas que permitan determinar el nivel de calidad del software y la cantidad de pruebas a realizar.



Pruebas sobre Casos de Uso

Las pruebas sobre CU



Utilizar los Casos de Uso y posiblemente los Contratos para elaborar los casos de prueba

- ▣ Numerar/nombrar el caso de prueba;
- ▣ Indicar el estado del sistema antes de la ejecución
- ▣ Entradas consideradas
- ▣ Salidas esperadas

Pruebas sobre CU



Para cada escenario de uso, elaborar casos de prueba que ejerciten:

- ▣ el curso normal del caso de uso;
- ▣ los cursos opcionales;
- ▣ considerar las excepciones que puedan ocurrir

Exigencia en las Pruebas



Exigencia Débil:

- ▣ Ejecutar al menos una vez cada acción condicionada de un curso normal
- ▣ Ejecutar cero, una y más veces una acción iterable de un curso normal
- ▣ Ejecutar al menos una vez cada opción
- ▣ Ejecutar al menos una vez cada excepción

Exigencia en las Pruebas



Exigencia Media:

- ▣ Además de las anteriores, realizar pruebas de frontera para cada parámetro de cada evento