Teoría 12: Ejecución y Prueba

La prueba de un programa es un chequeo mas minucioso que se hace sobre el **algoritmo** con la prueba de escritorio. Hacemos esto para saber si el programa soluciona el problema de la forma esperada.

Prueba Exhaustiva:

Se somete el programa a todas las posibles variaciones de entrada, sean validas o no.

La prueba ideal es imposible.

Hay que elegir bien los casos de prueba.

Fases de prueba

la prueba del programa se puede dividir en 3 etapas:

1. Prueba de unidad

Es probar uno a uno todos los modulos. Se puede dividir en **Fase informal** y **Fase sistemática**

La **fase informal** consiste en ir ejecutando el codigo (durante el desarrollo del programa) para conversese de que funciona.

En la **fase sistemática** se buscan fallos siguiendo algún criterio, como pueden ser, **caja blanca** o **caja negra**.

Caja Blanca

Cuando analizando el codigo se intenta hacerlo fallar, con animo de "probarlo todo".

Cobertura

es la medida porcentual del codigo que hemos cubierto con la prueba. **Tipos de cobertura**:

- **Cobertura de sentencias**: se trata de ejecutar todas, o la gran mayoría, de sentencias del programa.
- **Cobertura de ramas o decisción**: Es como la cobertura de sentencias pero mas minuciosa. Para tener una cobertura del 100% se debe satisfacer las decisiones al menos una vez por verdadero y otra por falso.
- **Cobertura de condición**: Analiza todas las situaciones posibles en condiciones logicas complejas (formadas por mas de una condición). Se prueban todos los valores de cada condición.

```
si (condición 1) o (condicion2)
entonces acciones1
sino acciones2
fsi
```

- **Cobertura de caminos**: Intenta cubrir todos los caminos del grafo de flujo de control. los ciclos dificultan esta tarea, por eso se intenta hacerlo ejecutandolo 0 veces, 1, o mas de una.

Aplicación Práctica

Se suele intentar alcanzar una cobertura cercana al 100% de sentencias. Es muy recomendable, aunque mas cosoto, conseguir una buena cobertura de caminos.

Se puede llevar a cabo con un depurador, un listado de modulos, un conjunto de pruebas, y un marcador para ir señalando por donde vamos pasando.

Caja Negra

Se centra en lo que se espera de un módulo, intentan encontrar casos en que el modulo no cumpla con su especificación.

El tester se limita a pasar datos como entrada y analizar las salidas.

Una técnica algebraica utilizada es la de "**Clases de equivalencia**"

Esta tecnica se divide en dos:

Parametros de entrada:

- Para un parametro de entrada que deba estar comprendido en un rango, hay 3 clases de equivalencias: **por debajo, en y por encima** del rango.
- Para un parametro que requeira un valor de entre los de un conjunto o un valor logico, aparecen dos clases de equivalencia.

Parametros de salida:

Son los mismos criteros que se aplica en **parametros de entrada**. Hay que intentar conseguir una salida para cada una de las equivalencias.

---

2. Prueba de Integración

Prueba la coherencia semantica entre los diferentes modulos, verificando que los modulos interactuen correctamente entre ellos.

Se analiza por etapas englobando progresivamente mas módulos, se puene empezar cuando ya tenemos unos pocos módulos.

Con un **diseño descendente** se empieza a probar desde los modulos mas generales a los mas basicos. esta pensado en terminos de la funcionalidad global.

Para cada prueba hay que simular el papel de los módulos inferiores que aún no estan disponibles.

Por el contrario, con un **diseño ascendente** se empieza por los modulos de base. De esta forma evitamos crear modulos "ficticios", pero se centra mas en el desarrollo que en las espectativas del cliente.

Estas pruebas se pueden plantear de el punto de vista estructural o funcional

Pruebas estructurales: es mas similar a caja blanca, debemos probar todas las llamadas entre modulos para lograr una buena cobertura.

Pruebas funcionales: es mas parecido a caja negra. Trata de encontrar fallos en las salidas de un modulo cuando este depende de otro.

3. Prueba de aceptación

Son las pruebas que hace el usuario.

Se hacen sobre el sistema completo y buscan que el progama haga lo que la especificacion y el manual indican que deberia hacer.

Muchas veces se pueden aplicar tecnicas demonimadas **pruebas alfa** y **pruebas beta**.

Pruebas alfa: Se invita al usuario al lugar de desarrolo y se realizan pruebas en un entorno controlado.

Pruebas beta: Se hacen luego de la prueba alfa. Se desarrollan en el lugar de trabajo del usuario, quedando este a solas con el programa y trata de encontrarle fallos, que luego infoirma a los desarrolladores.