Pruebas del Software:

Caja negra

Contenido

1.-Pruebas de caja negra

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software. **"Las pruebas no son opcionales. Un software sin pruebas es una bomba a punto de estallar"**

Las pruebas de caja negra se limitan a que el tester pruebe con *"datos" de entrada y estudie como salen*, sin preocuparse de lo que ocurre en el interior. Principalmente, se centran en módulos, la interfaz de usuario (pantalla, ficheros, canales de comunicación…) pero suelen ser útiles en cualquier módulo ya que todos o la mayoría tienen datos de entrada y salida que se pueden comprobar y verificar.

Como cualquier otra prueba, las de caja negra *se apoyan y basan en la especificación de requisitos y documentación funcional*, estos requisitos suelen ser más complejos que los internos, para ello realizaremos una "cobertura de especificación" que será muy recomendable para conseguir probar el mayor campo posible.

Vamos a tratar dos técnicas de caja negra con más profundidad:

- 1.-Partición de equivalencia
- 2.-Análisis de valores limite

2.-Partición de equivalencias

Se basa en la división del campo de entrada en un conjunto de clases de datos denominadas clases de equivalencia. Para ellos debemos identificar las entradas admitidas y no admitidas por nuestra aplicación, además de lo que debe suceder a continuación. Posteriormente se elaboran los casos de prueba que supondrán el caso de prueba de nuestra aplicación.

El diseño de caso de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia
2. Identificar los casos de prueba

2.1-Identificar clases de Equivalencia

Las clases de equivalencia son un conjunto de entrada que definen estados válidos y no válidos del sistema. Estas se obtienen de las condiciones descritas en las especificaciones.

Para cada entrada se identifican sus correspondientes clases de equivalencia válidas y no válidas. Para identificar fácil las claves de equivalencia válidas y no validas nos basamos en la siguiente tabla.

Condiciones de Entrada	**Número de clases de equivalencia válidas**	**Número de clase de equivalencia no válidas.**
Rango de Valores	1 clase que contemple los valores del rango	2 clases fuera del rango, una por encima y otra por debajo
Valor especifico	1 clase que contemple dicho valor	2 clases que representen un valor por encima y otro por debajo
Elementos de un conjunto tratados de forma diferente por el programa	1 clase de equivalencia por cada elemento	1 clase que represente un elemento fuera del conjunto
Condición lógica	1 que cumpla la condición	1 clase que no cumpla la condición.

Ejemplo de clases de equivalencia:

Rango de valores:

Identificar una clase válida y dos clases inválidas
Ej: Un contador entre 1 y 99

Clases Validas	Clases invalidas
1< números < 99	números <1
	números > 99

Valor especifico

Identificar una clase válida y dos clases inválidas
Ej: Grado de satisfacción de 1 a 10

Clases Validas	Clases invalidas
1< grado < 0	grado <1
	grado > 99

Condición lógica

Identificar una clase válida y una clase inválida
Ej: Acepta términos y condiciones legales.

Clases Validas	Clases invalidas
true	false

Elementos de un conjunto tratados de forma diferente por el programa

Se define una clase de equivalencia válida por cada miembro y una invalida**.**
Ej: El tipo de vehículo puede ser: autobús, taxi, camión, coche o moto.

Clases Validas	Clases invalidas
Una por cada uno	Todos los que no están en el conjunto por ejemplo: trailer, avión ..

Ej: Se pide un identificador entero positivo

Clases Validas	Clases invalidas
Enteros positivos	Enteros negativos
	Números con decimales
	Cadenas de texto
	Nulo

2.2-Identificar los Casos de prueba

El objetivo es minimizar el número de casos de prueba, así cada caso de prueba debe considerar tantas condiciones de entrada como sea posible.

No obstante, es necesario realizar con cierto cuidado los casos de prueba de manera que no se enmascaren faltas.

Así, para crear los casos de prueba a partir de las clases de equivalencia se han de seguir los siguientes pasos:

1. Asignar a cada clase de equivalencia un número único.

2. Hasta que todas las clases de equivalencia hayan sido cubiertas por los casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.

3. Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para cubrir una única clase no válida no cubierta.

2.3-Ejemplo: Banco

Considérese una aplicación bancaria, donde el usuario puede conectarse al banco por Internet y realizar una serie de operaciones bancarias. Una vez accedido al banco con las consiguientes medidas de seguridad (clave de acceso y demás), la información de entrada del procedimiento que gestiona las operaciones concretas a realizar por el usuario requiere la siguiente entrada:

- Código del banco. En blanco o número de tres dígitos. En este último caso, el primero de los tiene que ser mayor que 1.
- Código de sucursal. Un número de cuatro dígitos. El primero de ellos mayor de 0.
- Número de cuenta. Número de cinco dígitos.
- Clave personal. Valor alfanumérico de cinco posiciones.
- Orden. Este valor se introducirá según la orden que se desee realizar. Puede estar en blanco o ser una de las dos cadenas siguientes:
  - ✓ “Talonario”
  - ✓ “Movimientos”

En el primer caso el usuario recibirá un talonario de cheques, mientras que en el segundo recibirá los movimientos del mes en curso. Si este código está en blanco, el usuario recibirá los dos documentos.

2.3.1- Identificamos las particiones de equivalencia

Las clases de equivalencia derivadas para este programa. Cada una de las clases ha sido numerada para facilitar después la realización de los casos de prueba.

Condición de Entrada	Tipo	Clase Equivalencia Válida	Clase Equivalencia No Válida
Código banco	Lógica (puede estar o no) Si está es Rango	**1:** En blanco **2:** 100<= Código banco <= 999	**3:** Un valor no numérico **4:** Código banco < 100 **5:** Código banco > 999
Código sucursal	Rango	**6:** 1000 <= Código sucursal <= 9999	**7:** Código sucursal < 1000 **8:** Código sucursal >= 9999
Nº Cuenta	Valor	**9:** Cualquier número de cinco dígitos	**10:** Número de menos de cinco dígitos **11:** Número de menos de cuatro dígitos
Clave	Valor	**12:** Cualquier cadena de caracteres alfanuméricos de 5 posiciones	**13:** Cadena de menos de cinco posiciones **14:** Cadena de más de cinco posiciones
Orden	Conjunto, con comportamiento distinto	**15:**“” **16:**“Talonario” **17:** “Movimientos”	**18:** Cadena distinto de blanco y de las válidas **19:** “Talonarios” **20:** “Movimiento”

2.3.2- Identificamos los casos de prueba

En la tabla que mostramos a continuación identificamos los casos de prueba. En ella lo primero es identificar que caso de prueba es y que clase de equivalencia se va a realizar las siguientes columnas son los datos introducidos y por último el resultado esperado.

Nº Caso	Clase de equivalencia	Banco	Sucursal	Cuenta	Clave	Orden	Resultado
1	1, 6a, 9a, 12a,15	-	1000	00000	00000	“”	Todos los movimientos y talonario
2	2a, 6b, 9b,12b, 16	100	9999	99999	zzzzz	“Talonario”	Envío de talonario
3	2b, 6, 9, 12,17	999	1001	12345	Hyu56	“Movimientos”	Envió de movimientos
4	3, 6, 9, 12, 15	30A	1989	12347	Kuh98	“”	Código banco erróneo
5	4, 6, 9, 12, 15	99	1989	12347	Kuh98	“”	Código banco erróneo
6	5, 6, 9, 12, 15	1000	1989	12347	Kuh98	“”	Código banco erróneo
7	1, 7, 9, 12, 15	-	999	12347	Kuh98	“”	Código sucursal erróneo
8	1, 8, 9, 12, 16	-	10000	12345	Hyu56	“Talonario”	Código sucursal erróneo
9	1, 6, 10, 12,16	-	2345	9999	Jkgy5	“Talonario”	Número cuenta erróneo
10	1, 6, 11, 12,16	-	7863	100000	Jkgy5	“Talonario”	Número cuenta erróneo
11	1, 6, 9, 13, 16	-	6754	89765	Jut8	“Talonario”	Clave errónea
12	1, 6, 9, 14, 16	-	9998	89765	Jut890	“Talonario”	Clave errónea
13	1, 6, 9, 12, 18	-	8765	89765	Ghy78	988	Orden errónea
14	1, 6, 9, 12, 19	-	7654	89765	Ghy78	“Talonarios”	Orden errónea
15	1, 6, 9, 12, 20	-	8769	89765	Ghy78	“Movimiento”	Orden errónea

Una vez diseñados los casos de prueba el siguiente paso sería llevarlas a cabo en el programa y comprobar el resultado esperado con el que la aplicación proporciona.

3.-Análisis de valores límite.

Esta técnica se basa en la evidencia experimental de que los errores suelen aparecer con mayor probabilidad en los extremos de los campos de entrada. Esta técnica se enfoca en la identificación de los casos de prueba asociados con los valores límites del dominio de la función tanto de entrada como de salida.

Por ejemplo:

```
if (usuario.getEdad() > 18) {
      permitirAcceso(usuario);
}
```

```
if (usuario.getEdad() >= 18) {
      permitirAcceso(usuario);
}
```

En muchas ocasiones los errores se cometen en los limites de las condiciones establecidas en la aplicación es por ello. Es por ello que es necesario diseñar casos de prueba que examinen dichos valores límite.

Por lo tanto, el análisis de valores límite complementa la técnica de partición de equivalencia de manera que:

- En lugar de seleccionar cualquier caso de prueba de las clases válidas e inválidas, se eligen los casos de prueba en los extremos.
- En lugar de centrarse sólo en el dominio de entrada, los casos de prueba se diseñan también considerando el dominio de salida.

Ejemplo:

Permitir acceso a personas que tengan 18 años.

N.º Caso	Edad	Resultado
1	17	Denegado
2	18	Acceso
3	19	Acceso

Ventajas de la técnica:

- La técnica reduce el número de casos de pruebas que deben ser creados y ejecutados.
- Esta técnica permite elegir un subconjunto de las pruebas que son eficiente y eficaces en encontrar no conformidades.
- La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen.