Testing de Caja Negra

Problemas asociados al testing

- Sobre el input:
  ¿Con qué datos probar?
  El objetivo es, con la menor cantidad de casos de prueba, considerar la mayor cantidad de escenarios.
- Sobre el resultado esperado:
  ¿Cómo saber qué se espera como resultado esperado?
  - A partir del código fuente.
  - A partir de la especificación o contrato
- Sobre el resultado obtenido:
  ¿Cómo saber cuál fue el resultado obtenido?
  - A veces puede no ser trivial ver o inspeccionar el resultado obtenido (cuanto más complejo el sistema, más difícil)

Criterios de caja negra o funcionales

Los criterios de caja negra, también conocidos como criterios funcionales, son métodos utilizados en el testing de software para diseñar y seleccionar pruebas basadas únicamente en la especificación de los requisitos funcionales del sistema, sin considerar su implementación interna. Estos criterios se centran en probar el comportamiento externo del software y validar que cumpla con los resultados esperados.

Los datos de test se derivan a partir de la descripción del programa sin
conocer su implementación

Método de partición de categorías

El método de partición de categorías es una técnica utilizada en el testing de software para seleccionar un conjunto representativo de casos de prueba a partir de un dominio de entrada. Este método se basa en dividir el conjunto de datos de entrada en categorías o clases y seleccionar casos de prueba que representen cada una de estas categorías.

Es aplicable a especificaciones formales, semiformales e inclusive, informales.

Pasos

1. Listar todos los problemas que queremos testear.
2. Elegir uno en particular.

3. Identificar sus parámetros o las relaciones entre ellos que condicionan su comportamiento. Los llamaremos genéricamente factores.
4. Determinar las características relevantes (categorías) de cada factor.
5. Determinar elecciones (choices) para cada característica de cada factor.
6. Clasificar las elecciones: errores, únicos, restricciones, etc.
7. Armado de casos, combinando las distintas elecciones determinadas para cada categoría, y detallando el resultado esperado en cada caso.
8. Volver al paso 2 hasta completar todas las unidades funcionales.

Ejemplo

Paso 1: Descomponer la solución informática en unidades funcionales.
Consiste en enumerar todas las operaciones, funciones, funcionalidades, problemas que se probarán.

$$\begin{array}{l}\text{problema } esPermutacion(s1, s2 : seq\langle T\rangle) : \text{Bool } \{ \\ \quad \text{asegura: } \{res = true \leftrightarrow ((\forall e : T)(cantidadDeApariciones(s1, e) = \\ cantidadDeApariciones(s2, e)))\} \\ \} \\ \text{problema } cantidadDeApariciones(s : seq\langle T\rangle, e : T) : \mathbb{Z} \ \{ \\ \quad \text{requiere: } \{e \in s\} \\ \quad \text{requiere: } \{|s| > 0\} \\ \quad \text{asegura: } \{res = \sum_{i=0}^{|s|-1} (\text{if } s[i] = e \text{ then 1 else 0 fi})\} \\ \}\end{array}$$

En este caso:

- esPermutacion
- cantidadDeApariciones

Paso 2: Elegir una unidad funcional
Lo ideal es llegar a testear todas las unidades funcionales. Un buen criterio, es empezar por aquellas que son utilizadas por otras.

En este caso:
cantidadDeApariciones

Paso 3: Identificar factores
Esto pueden ser los parámetros del problema a testear (si el sistema es más complejo... podrían ser otros factores).

En este caso:

- Tomamos T como Z (el parámetro va a ser entero)

- s : seq⟨Z⟩
- e : Z

Paso 4: Determinar categorías
Las categorías son distintas características de cada factor, o características que relacionan diferentes factores, y que tienen influencia en los resultados. Son el resultado del análisis de toda la información disponible sobre la funcionalidad a testear.

En este caso, para cada parámetro podemos determinar las siguientes caracteristicas:

- s: seq⟨Z⟩
  - ¿Tiene elementos?
- e : Z
  - En este caso, para e no se distingue ninguna característica interesante
- Relación entre s y e
  - ¿Pertenece e a s?

Paso 5: Determinar elecciones
Se trata de buscar los conjuntos de valores donde se espera un comportamiento similar. Se basa en las especificaciones, la experiencia, el conocimiento de errores.

En este caso, para cada categoría, determinamos sus elecciones o choices:

- s: seq⟨Z⟩
  - ¿Tiene elementos?
    -Sí
    -No
- e : Z
- Relación entre s y e
  - ¿Pertenece e a s?
    -Sí
    -No

Paso 6: Clasificar las elecciones
Se trata de identificar algunas propiedades o restricciones de las elecciones en el marco de la unidad funcional.
Las clasificaciones más comunes son:

- Error: Se clasificarán como error aquellas elecciones que por sí mismas determinen que como resultado de la ejecución el sistema debe detectar un error o que no está definido su comportamiento.
- Único: Son aquellas elecciones que no necesitan combinarse con ninguna otra elección para determinar el resultado esperado.

Para cada elecciones o choices, analizamos si debemos clasificarlo especialmente:

- s: seq⟨Z⟩
  - ¿Tiene elementos?
    -Sí
    -No [ERROR]
- e : Z
- Relación entre s y e
  - ¿Pertenece e a s?
    -Sí
    -No [ERROR]

Paso 7: Armar los casos de test
Finalmente, se combinarán las distintas elecciones de las categorías consideradas generando los distintos casos de test. *Cada combinación es un caso de test*, el cual deberá tener identificado con claridad su resultado
esperado.

Las elecciones marcadas como ERROR y UNICO, suelen no ser
combinables (recortando así la cantidad de combinaciones a realizar).
Las elecciones RESTRICCION, condicionan las combinaciones posibles.

Por cada caso, debemos describir su resultado esperado: es importante
indicar si el resultado será un posible resultado correcto u esperable o un error o comportamiento indefinido. Los casos de prueba definido serán una herramienta que eventualmente otra persona pueda ejecutar los test: eligiendo datos concretos y comparando el resultado obtenido con el esperado.

La tabla es una representación gráfica y práctica de los casos.

- Suele ocurrir, que las primeras columnas son siempre de aquellas elecciones que tienen errores, unicos o restricciones entre sus posibles valores: porque descartan casos hacia la derecha.

En nuestro ejemplo, deberemos combinar:
▶ ¿s tiene elementos?: Si
▶ ¿s tiene elementos?: No

▶ ¿e pertenece a s?: Si
▶ ¿e pertenece a s?: No

Tenemos 4 elecciones posibles a combinar
¿Nos interesan todas las combinaciones? ¿Cuántos casos de test tenemos?
Sólo nos quedan 3 casos interesantes (nos ahorramos 1).

Característica	¿s tiene elementos?	¿e pertenece a s?	Resultado esperado	Comentario
Caso 1: S sin elementos	No	-	ERROR: no está espeficado que sucede en este caso.	Como la elección No es un ERROR, no importa el valor
Caso 2: E no pertenece	-	No	ERROR: no está espeficado que sucede en este caso.	Como la elección No es un ERROR, no importa el valor
Caso 3: S tiene elementos y e Pertenece	Si	Si	OK: el resultado obtenido debe ser igual a la cantidad de	

Ejemplo Paso 6 (ÚNICO)

Único: Son aquellas elecciones que no necesitan combinarse con ninguna otra elección para determinar el resultado esperado.

$$
\begin{array}{l}
\text{problema } \mathit{siElPrimeroEsCincoHaceOtraCosa}(x : \mathbb{Z}, y : \mathbb{Z}, z :) : \mathbb{Z} \ \{ \\
\quad \text{requiere: } \{x = 5 \longrightarrow z \neq 0\} \\
\quad \text{asegura: } \{x = 5 \longrightarrow \mathit{res} = x + y/z\} \\
\quad \text{asegura: } \{x \neq 5 \longrightarrow \mathit{res} = x + y \\
\}
\end{array}
$$

- Factor X
  - Característica: Valor.
    -Elección: Igual a 5 **[Restricción]**
    -Elección: Distinto que 5.
- Factor Y
- Factor Z
  - Característica: ¿Es distinto de 0?
    -Elección: Sí.
    -Elección: No. **Solo si x=5 hay [Restricción]**