ETS-UT3-5. Pruebas de Caja negra. Análisis de valores límite

Análisis de valores límite

Esta metodología de caja negra se basa en el hecho heurístico de que los errores tienden a producirse con más probabilidad en los **límites** o **extremos** de los campos de entrada.

Está técnica complementa a la anterior de las clases de equivalencia de forma que los casos de prueba elegidos toman valores justo por encima y por debajo de los extremos de las clases de equivalencia.

Además, no solo se miran las **condiciones de entrada**, sino que también se exploran las **condiciones de salida** definiendo las clases de equivalencia de salida.

Esta heurística se aplica a los casos en los que los valores de entrada/salida se corresponden a **rango** de válores o **número** de valores. Para el caso de que los valores sean de tipo lógico o un conjunto de valores se aplican las mismas reglas que para las clases de equivalencia.

Directrices

1. Rangos para condición de entrada

Si una condición de entrada especifica un **rango de valores** se deben diseñar casos de prueba para los **límites** del rango y para valores justo por **encima** y por **debajo** del rango

Ejemplo: días de la semana en formato numérico; números del 1 al 7 ambos inclusive. Se generan casos de prueba para:

- Caso de prueba dia = 1 | Clase válida
- Caso de prueba dia = 7 | Clase válida
- Caso de prueba dia = 0 | Clase no válida
- Caso de prueba dia = 8 | Clase no válida

2. Número de valores para condición de entrada

Si una condición de entrada especifica un número de valores, se deben diseñar casos de prueba que comprueben los valores máximo y mínimo; un valor por debajo del mínimo y un valor por encima del máximo.

Ejemplo El código del departamento debe estar entre 1 y 100. Se generan casos de prueba para los valores 0,1, 100 y 101.

3. Rango para condición de salida

Aplicar regla 1 para la condición de salida.

Ejemplo dependiendiendo del tipo de cliente se obtiene un tipo de descuento de tipo **real** que está en 10 y el 50%. Probamos con clientes que generen salidas con descuento de 9,99%, 10%, 50% y 50,01%

4. Número de valores para condición de salida

Aplicar regla 2 para la condición de salida.

Ejemplo un programa genera de salida una tabla con valores enteros entre 1 y 10 temperaturas. Se generan casos de prueba para obtener 0, 1, 10 y 11 temperaturas.

Ejemplos

1. Casos de prueba para los siguientes elementos según las condiciones de entrada o salida:

	Condiciones de entrada/salida	**Casos de prueba**
Código	Entero de 100 a 999	Valores: 99, 100, 999, 1000
Puesto	cadena de hasta 4 caracteres	Longitud caracteres: 0, 1, 4, 5
Antigüedad	De 0 a 25 años real	Valores: -0.1, 0, 25, 25.1
Horas semanales	de 0 a 60	Valores: -1, 0, 60, 61
Fichero de entrada	Tiene de 1 a 100 registros	Para leer 0, 1, 100 y 101 registrso
Fichero de salida	Podrá tener de 0 a 10 registros	PAra generar 0, 10 y 11 registros (no se puede generar -1 registro)

2. Supuesto de clases de equivalencia

Se va a realizar la entrada de datos de un empleado desde un formulario. Se definen 3 campos de entrada:

- **Empleado**: número de 3 dígitos que no empiece por 0. Campo de entrada numérico.
- **Departamento**: en blanco o número de 2 dígitos almacenado en variable alfanumérica. Entrada se teclea manualmente.
- **Oficio**: Analista, Diseñador, Programador o Elige oficio. Entrada se elige de lista desplegable.

Si la entrada es correcta el programaa asigna un salario según estas normas:

- Oficio Analista -> **S1** = 2500
- Oficio Diseñador -> **S2** = 1500
- Oficio Programador -> **S3** = 2000

Si la entrada es incorrecta el programa muestra mensaje indicando la entrada incorrecta:

- **ER1** si el **Empleado** no es correcto.
- **ER2** si el **Departamento no es correcto.
- **ER3** si no se ha elegido **Oficio**

Tenemos la siguiente función de Python, de la que no disponemos del código, que recibe los parámetros de entrada y debe generar la salida correspondiente:

```
def salario_empleado(num_empleado, num_departamento, oficio):
    """
    Dada información de empleado devuelve su salario
    :param num_empleado: int
    :param num_departamento: str
    :param oficio: str
    :return int
    """
```

1. Empezamos creando tabla que represente las clases de equivalencia

Para representar las clases de equivalencia para cada condición de entrada se puede usar una tabla de la siguiente forma:

Condición de entrada	**Clase de equivalencia**	**Valores límite válidos**	**COD**	**Valores límite no válidos**	**COD**
Empleado	Rango	100, 999	V1, V2	99, 1000	NV1, NV2
Departamento	Lógica	En blanco	V3	0a	NV3
Departamento	Valor	00, 99	V4, V5	9, 100	NV4, NV5
Oficio	Miembro conjunto	"Programador"	V6	Oficio = "Elige oficio"	NV6
		"Analista"	V7		
		"Diseñandor"	V8		

2. A partir de esa tabla se generan los **casos de prueba**:

Caso de prueba	**Clases de equivalencia**	**Empleado**	**Departamento**	**Oficio**	**Resultado esperado**
CP1	V1, V3, V6	100		Programador	S3
CP2	V2, V4, V7	999	00	Analista	S1
CP3	V1, V5, V8	100	99	Diseñador	S2
CP4	NV1, V3, V6	99		Programador	ER1
CP5	NV2, V4, V7	1000	00	Analista	ER1
CP6	V2, NV3, V8	999	0a	Diseñador	ER2
CP7	V1, NV4, V6	100	9	Programador	ER2
CP8	V2, NV5, V7	999	100	Analista	ER2
CP9	V1, V4, NV6	100	00	Elige oficio	ER3

Recuerda que debemos elegir los casos de prueba de forma que generemos combinaciones en las que se seleccione, al menos 1 vez, todos los valores válidos y los no válidos. Así mismo, para los valores no válidos se deben aplicar de uno en uno.

Para este supuesto no se generan casos de prueba para las salidas ya que estas no son ni un rango ni un número de valores, sino miembros de un **conjunto**.

3. Una vez que tenemos los casos de prueba. En caso de usar **Doctest** para los casos de prueba estos podrían ser:

```
def salario_empleado(num_empleado, num_departamento, oficio):
    """
    Dada información de empleado devuelve su salario
    :param num_empleado: int
    :param num_departamento: int
    :param oficio: str
    :return int

    CP1
    >>> salario_empleado(100, "", "Programador")
    2000

    CP2
    >>> salario_empleado(999, "00", "Analista")
    2500

    CP3
    >>> salario_empleado(100, "99", "Diseñador")
    1500

    CP4
    >>> salario_empleado(99, "", "Programador")
    'Número de empleado incorrecto'

    CP5
    >>> salario_empleado(100, "00", "Analista")
    'Número de empleado incorrecto'

    CP6
    >>> salario_empleado(999, 0a, "Programador")
    'Número de departamento incorrecto'

    CP7
    >>> salario_empleado(100, '9', "Programador")
    'Número de departamento incorrecto'

    CP8
    >>> salario_empleado(999, "100", "Analista")
    'Número de departamento incorrecto'

    CP9
    >>> salario_empleado(100, "00", "Elige oficio")
    'Debes elegir un oficio'
    """
```

Recursos

- Particiones de equivalencia: Pruebas de caja negra - Educando con TIC
- Valores límite: Pruebas de caja negra - Educando con TIC