Enunciado

Se desea efectuar el testing funcional de un programa que ejecuta transferencias entre cuentas bancarias. El programa recibe como parámetros la cuenta de origen, la de cuenta de destino y el monto a transferir.

Se aplican las siguiente consideraciones:

- Las cuentas pueden ser Cajas de Ahorro o Cuentas Corrientes.
- Si la cuenta de origen es una Caja de Ahorro, el monto a transferir no puede ser mayor al saldo disponible de la cuenta.
- Si la cuenta de origen es una Cuenta Corriente, el monto a transferir no puede dejar la cuenta con un saldo menor a su límite preacordado de descubierto.
- Ambas cuentas deben estar activas.

Los siguientes puntos caracterizan los restantes datos de entrada:

- El monto a transferir debe ser un número positivo.
- Las cuentas deben pertenecer al banco.

Derive casos de test funcional a partir del enunciado anterior identificando las clases de equivalencia para los parámetros de entrada.

La técnica

Llamamos dominio de una funcionalidad, al conjunto de todos sus posibles input. Llamamos partición a un subconjunto de ese dominio.

La idea básica del método de partición de categorías es dividir el dominio en particiones que lo cubran sin solaparse, y poder realizar el testing utilizando datos representativos de esas particiones

Es importante destacar que este método aplica tanto a sistemas con especificaciones formales, semiformales o informales

Los pasos para el diseño de los casos de prueba con el método de partición en categorías son:

1. Elegir una funcionalidad que puedan testearse en forma independiente.
2. Identificar sus parámetros u otros objetos del ambiente que pueden afectar su funcionamiento. Los llamaremos genéricamente factores.
3. Determinar categorías de cada factor. Las categorías son distintas características de cada factor, o características que relacionan diferentes factores, y que tienen influencia en los resultados (outputs) que producirá la funcionalidad.
4. Determinar elecciones (choices) para cada característica de cada factor. Se trata de buscar los conjuntos de valores donde se espera un comportamiento similar (ver la lista de heurísticas).
5. Clasificar las elecciones:
    - Errores: Se clasificarán como error aquellas elecciones que por si mismas determinen que como resultado de la ejecución, el sistema debe detectar un error. En general, no se deseará combinar con todas las posibles elecciones del resto de las categorías. Lo voy a denotar como [ERROR]
    - Únicos: Se clasificarán como únicos aquellas elecciones a las que se pretende que queden incluidas en un único caso de test, sin combinarse con todas las posibles elecciones del resto de las categorías. En estos casos se deberá elegir en forma arbitraria una elección para las otra categorías en el caso de test, o se podrá no indicar nada para ellas quedando a criterio de quien genere los datos de test (esto es muy probable que no sea lo deseado). Lo voy a denotar como [UNICO]
    - Restricciones: Se clasificarán con restricciones aquellas elecciones que sólo quieran combinarse con otras elecciones que cumplan determinada propiedad. Para estos casos, denoto a la propiedad con [Propiedad "nombre"] y a la restricción con [SI "nombre"]
6. Armado de casos, combinando las distintas elecciones determinadas para cada categoría, y detallando el resultado esperado en cada caso. Tener en cuenta que:
    - en la combinación se considerarán las restricciones que puedan surgir de las clasificaciones que se pudiesen haber establecido.
    - Al identificar el resultado esperado no se trata de decir OK u ERROR, sino de dar un descripción detallada de cuál es el error que debe detectar el sistema o cuál es la poscondición de la ejecución exitosa de la funcionalidad, de acuerdo a la información que se disponga. Notar que el resultado esperado se va a comparar con la ejecución real con los datos de prueba, y el tester tiene que estar en

condiciones de verificar que el comportamiento cumplió con las expectativas. Si se pusiera "OK" como resultado esperado ¿qué es lo que debería validarse al momento de ejecutar el test? ¿qué el sistema diga OK? Por eso es importante que se detalle completamente todos los outputs de la unidad funcional, ya sean parámetros de salida, resultado (si es una función), modificación del estado del sistema o de objetos del ambiente.

Una vez terminado el diseño de los casos de prueban, se está en condiciones de continuar el proceso de testing por su paso siguiente (preparar datos de prueba).

Algunas heurísticas a tener en cuenta en la determinación de elecciones:

a. Se deben generar particiones completas, aunque quizás para algunas de ellas no se puedan encontrar posteriormente datos de prueba.
b. Si una categoría especifica un intervalo de valores incluidos, puede ser conveniente identificar una clase válida (los que están dentro del intervalo) y dos inválidas (los menores que el intervalo y los mayores que el intervalo)
c. Se recomienda poner especial énfasis en lo que se llaman "situaciones de borde", ya que gran cantidad de defectos ocurren sobre los mismos. Por ejemplo, si una categoría fuese un valor numérico, y se viese un intervalo relevante [a, b], es importante tratar de forma diferenciada a los valores a-1, a, b, b+1.
d. Si una categoría especifica un conjunto de valores (por ejemplo, un enumerado), y hay razones para pensar que se manejan en forma distinta dentro del programa, se recomienda tener una elección por cada valor válido que esa categoría puede tomar, y también una elección para un valor inválido
e. Si una categoría especifica una situación que debe ocurrir, identificar una elección válida (ocurre lo que debe ocurrir) y una inválida (no ocurre lo que debe ocurrir).
f. Siempre que se pueda, agregar elecciones para datos que sean de diferente tipo que el esperado (aunque quizás después no haya forma de generar datos para ello)
g. Prestar atención a la multiplicidad de las relaciones especificadas en modelos conceptuales o modelos de datos, ya que ellas definen reglas de dominio que deben ser probadas.
h. Si existen modelos que definan el ciclo de vida de entidades o clases conceptuales (ya sean FSMs, diagramas de estados u otros), deberán probarse tanto transiciones válidas como inválidas.
i. Al seleccionar datos de prueba para un caso de test, se elegirá sólo un valor por cada elección. Si se considera importante que se elijan dos valores representativos, deberían detallarse distintas elecciones que contengan a cada uno de esos dos valores representativos.

Resolución

1. Determinar la funcionalidad a testear

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**

2. Determinar factores

En un principio detectamos cuenta de origen, cuenta de destino y monto.

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**
Cuenta de origen (CO)			
Cuenta de destino (CD)			
Monto (M)			

3. Determinar categorías

Identificamos

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor/es	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**
Cuenta de origen (CO)	Tipo		
	Estado		
Cuenta de destino (CD)	Estado		
	Tipo		
Monto	Signo		
Monto / Saldo	Relación entre monto y saldo		
Monto / Saldo / Limite de descubierto	Relación entre M vs S vs LD		
CO / Cuentas del Banco	CO pertenece al Banco?		
CD / Cuentas del Banco	CD pertenece al Banco?		

4. Determinar elecciones ("choices") para cada categoría

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor/es	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**
Cuenta de origen (CO)	Tipo	Caja de Ahorro	
		Cuenta Corriente	
	Estado	Activa	
		Inactiva	
Cuenta de destino (CD)	Estado	Activa	
		Inactiva	
	Tipo	Caja de Ahorro	
		Cuenta Corriente	
Monto	Signo	<0	
		=0	
		>0	
Monto / Saldo	Relación entre monto y saldo	M>S	
		M=S	
		M<S	
Monto / Saldo / Limite de descubierto	Relación entre M vs S vs LD	S-M<LD	
		S-M=LD	
		S-M>LD	
CO / Cuentas del Banco	CO pertenece al Banco?	Si	
		No	
CD / Cuentas del Banco	CD pertenece al Banco?	Si	
		No	

Tener en cuenta que, para testear el programa, se seleccionará un valor representativo de cada una de las clases de equivalencia identificadas, asumiendo que los resultados que se obtendrán serán los mismos para cualquiera de los elementos de la clase. Es decir, se espera que, si la prueba con uno de los elementos de la clase da error, entonces, la prueba con los demás elementos de la clase dará el mismo error. Y, si la prueba con uno de los elementos da un resultado correcto, se espera el mismo resultado en la prueba de cualquiera de los demás elementos.

5. Clasificar las elecciones: errores, únicos, restricciones

Si no restringiéramos la combinación de las elecciones, ¿cuántos casos de test tendríamos que generar con la tabla del punto anterior?

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor/es	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**
1.Cuenta de origen (CO)	1.1.Tipo CO	Caja de Ahorro	[Propiedad "la CO es CA"]
		Cuenta Corriente	[Propiedad "la CO es CC"]
	1.2.Estado CO	Activa	
		Inactiva	[ERROR]
2.Cuenta de destino (CD)	2.1.Estado CD	Activa	
		Inactiva	[ERROR]
	2.2.Tipo CD	Caja de Ahorro	
		Cuenta Corriente	

Funcionalidad: Transferencias de dinero entre cuentas bancarias			
Factor/es	**Categoría**	**Elección (Choice)**	**Clasificación de la elección**
3.Monto (a transferir)	3.1.Signo M	<0	[ERROR]
		=0	[ERROR]
		>0	
4.Monto / Saldo CO	4.1.Relación entre Monto vs Saldo CO (M vs S)	M>S	[SI "la CO es CA"] [ERROR]
		M=S	[SI "la CO es CA"]
		M<S	[SI "la CO es CA"]
5.Monto / Saldo / Limite de descubierto	5.1.Relación respecto del límite de descubierto (M vs S vs LD)	S-M<LD	[SI "la CO es CC"] [ERROR]
		S-M=LD	[SI "la CO es CC"]
		S-M>LD	[SI "la CO es CC"]
6.CO / Cuentas del Banco	6.1.CO pertenece al Banco?	Si	
		No	[ERROR]
7.CD / Cuentas del Banco	7.1.CD pertenece al Banco?	Si	
		No	[ERROR]

6. Armar los casos de test

¿Cuántos casos nos deberían quedar con la clasificación que hicimos?
2 x 1 x 1 x 2 x 1 x 2 x 1 x 1 + 8 = 16

A partir de la tabla de elecciones generada, armamos los casos de test combinando los choices cuando corresponda.

Esta tabla de casos de test es un posible conjunto de casos de test para la funcionalidad en cuestión.

Casos de prueba para la funcionalidad: Transferencias de dinero entre cuentas bancarias										
Id	**6.1**	**7.1**	**1.2**	**2.1**	**3.1**	**1.1**	**5.1**	**4.1**	**2.2**	
	CO ∈ B	**CD ∈ B**	**estado CO**	**estado CD**	**M**	**tipo CO**	**M vs S vs LD**	**M vs S**	**tipo CD**	**Resultado Esperado**
1	No	-	-	-	-	-	-	-	-	**Error**, la CO no es cuenta del Banco.
2	Sí	No	-	-	-	-	-	-	-	**Error**, la CD no es cuenta del Banco.
3	Sí	Sí	Inactiva	-	-	-	-	-	-	**Error**, la CO está inactiva.
4	Sí	Sí	Activa	Inactiva	-	-	-	-	-	**Error**, la CD está inactiva.
5	Sí	Sí	Activa	Activa	M<0	-	-	-	-	**Error**, el monto debe ser un número positivo.
6	Sí	Sí	Activa	Activa	M=0	-	-	-	-	**Error**, el monto debe ser un número positivo.
7	Sí	Sí	Activa	Activa	M>0	CC	S-M<LD	no aplica	-	**Error**, el monto a transferir no puede dejar la cuenta con un saldo menor a su límite preacordado de descubierto.
8	Sí	Sí	Activa	Activa	M>0	CA	no aplica	M>S	-	**Error**, el monto a transferir no puede ser mayor al saldo disponible de la cuenta.
9	Sí	Sí	Activa	Activa	M>0	CC	S-M=LD	no aplica	CC	**OK,** la transferencia se realiza exitosamente de una CC a otra CC. La CC de origen queda con su saldo igual al límite preacordado de descubierto (+ ver nota al pie)
10	Sí	Sí	Activa	Activa	M>0	CC	S-M=LD	no aplica	CA	**OK,** la transferencia se realiza exitosamente de una CC a una CA. La CC de origen queda con su saldo igual al límite preacordado de descubierto (+ ver nota al pie)
11	Sí	Sí	Activa	Activa	M>0	CC	S-M>LD	no aplica	CC	**OK,** la transferencia se realiza exitosamente de una CC a otra CC. (+ ver nota al pie)
12	Sí	Sí	Activa	Activa	M>0	CC	S-M>LD	no aplica	CA	**OK,** la transferencia se realiza exitosamente de una CC a una CA. (+ ver nota al pie)
13	Sí	Sí	Activa	Activa	M>0	CA	no aplica	S=M	CC	**OK,** la transferencia se realiza exitosamente de una CA a una CC. El saldo de la CA de origen queda en 0. (+ ver nota al pie)
14	Sí	Sí	Activa	Activa	M>0	CA	no aplica	S=M	CA	**OK,** la transferencia se realiza exitosamente de una CA a otra CA. El saldo de la CA de origen queda en 0. (+ ver nota al pie)
15	Sí	Sí	Activa	Activa	M>0	CA	no aplica	S<M	CC	**OK,** la transferencia se realiza exitosamente de una CA a una CC. (+ ver nota al pie)
16	Sí	Sí	Activa	Activa	M>0	CA	no aplica	S<M	CA	**OK,** la transferencia se realiza exitosamente de una CA a otra CA. (+ ver nota al pie)

Nota:
Que la transferencia sea exitosa significa que se deberá verificar que:

- el saldo de la cuenta de origen debe ser igual al saldo que tenía menos el monto (S-M).
- el saldo de la cuenta de destino debe ser igual al que tenía antes de la transferencia más M.

Al armar la tabla de casos de test, no hay obligación de poner las categorías en orden de columnas similar al que fueron definidas, se trata de ordenarlas de manera que la tabla sea lo más legible posible. Un criterio que muchas veces es de utilidad es tratar de poner en primer lugar las categorías/elecciones que involucren casos erróneos, de manera que la tabla a veces toma forma de una matriz diagonal, o una matriz que tiene las filas completas en su sector inferior. Tener en cuenta que no es un objetivo que el contenido de la tabla la muestre como una matriz diagonal, a veces por forzar esta condición se pueden dejar casos sin explicitar. En este caso, agregamos una fila con un id numérico a cada categoría simplemente para que sea más legible saber cuál categoría es el nombre corto de la columna. Esto tampoco es obligatorio de hacer.

Algunas consideraciones/aclaraciones:

Durante la clase práctica se armó la tabla de choices con una partición adicional en CO para Saldo con 3 choices (<0, >0, =0). Sin embargo al armar la tabla de casos resultante, quedaban muchos casos de test para los cuales no iban a existir datos de test (como ser: que M>0, que M>S y que S<0). Esto se dio porque por un lado teníamos la categoría saldo, por el otro monto y por el otro la relación entre ambos. Pero a su vez, la relación entre saldo y monto también está dada por la combinación a la hora de armar los casos. Es por eso que no se eliminó la partición Saldo.

Notar que estamos combinando casos de test por tipo CO tipo CD y relaciones entre monto y saldo. Se podría considerar que no es necesario combinar de esa manera ya que es esperable que el comportamiento de la transferencia pueda verse afectado segun los tipos de CO y CD, pero no tiene sentido que también varíe dependiendo del tipo de CO, tipo de CD y la relación entre monto, saldo y LD. En ese caso se agregarían restricciones para eliminar combinaciones redundantes.

La técnica propone 6 pasos, sin embargo el proceso es iterativo, es decir, al construir la tabla de casos de prueba se pueden identificar casos redundantes y volver al paso anterior para agregar restricciones y volver a generar los casos de prueba.

¿Se le ocurre alguna forma de restringuir que los choices de tipo CD solo se combinen cuando tipo CO es CC y cuando tipo CO es CA sin importar las relaciones entre monto, saldo y LD?