UTN.BA FACULTAD REGIONAL BUENOS AIRES

SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

www.sceu.frba.utn.edu.ar/e-learning

Professional Testing Master

Presentación:

En esta Tercera Unidad del curso, nos adentraremos en los detalles de la estrategia de testing y métodos de diseño de casos de prueba para el caso de las aplicaciones orientadas a objetos (OO).

Unidad 3:

Testing de aplicaciones orientadas a objetos

Objetivo:

Al terminar la Unidad los participantes:

- ❑ Habrán ensanchado la visión del testing para incluir la revisión de los modelos de requerimientos y de diseño.
- ❑ Estarán en condiciones de diseñar pruebas unitarias efectivas aplicadas a clases y objetos.
- ❑ Conocerán nuevas estrategias para llevar a cabo las pruebas de integración.

Contenido

1. Testing de modelos de análisis y diseño orientados a objetos
2. Estrategias de testing orientadas a objetos
3. Métodos de testing orientados a objetos
4. Métodos de testing aplicables a nivel de clase
5. Diseño de casos de prueba inter-clase

Introducción

La naturaleza de los programas OO cambia las estrategias y las tácticas de las pruebas.

Se debe ampliar la definición de prueba para incluir la revisión de los modelos de análisis y de diseño orientado a objetos.

Prueba de modelos de análisis y diseño

En esta sección:

- ❑ Exactitud de los modelos AOO y DOO
- ❑ Consistencia de los modelos orientados a objetos

Prueba de modelos de análisis y diseño

Exactitud de los modelos de análisis y de diseño orientados a objetos (AOO y DOO respectivamente)

- Exactitud semántica: Las relaciones de clase se evalúan para determinar si reflejan con precisión conexiones de objetos en el mundo real. Los casos de uso son invaluables para este fin.

Consistencia de los modelos AOO y DOO

Un modelo de análisis o diseño inconsistente tiene representaciones en una parte del modelo que no se reflejan de manera correcta en otras partes.

Para evaluar la consistencia, debe examinarse cada clase y sus conexiones con otras clases.

A fin de facilitar esta actividad, puede usarse el modelo clase-responsabilidad-colaboración (CRC), que se compone de fichas índice CRC.

Consistencia de los modelos AOO y DOO (cont.)

Para evaluar el modelo de clases se debe usar el modelo CRC para:

- Verificar que las colaboraciones implicadas por el modelo de requerimientos se reflejan de manera adecuada en el CRC.
- Verificar que cada responsabilidad invocada es provista por la clase colaboradora y viceversa.
- Verificar si responsabilidades requeridas juntas frecuentemente pueden agruparse en una sola.

Estrategias de prueba orientadas a objetos

En esta sección:

- ❑ Prueba unitaria en el contexto OO
- ❑ Prueba de integración en el contexto OO
- ❑ Prueba de validación en un contexto OO

Prueba unitaria en el contexto OO

La unidad de prueba más pequeña es la clase, y no el módulo o procedimiento.

La prueba de clase para el software OO es el equivalente a la prueba unitaria en el software tradicional.

La prueba de clase se activa mediante las operaciones encapsuladas por la clase y por el comportamiento de estado de la misma.

Prueba de integración en el contexto OO

- La unidad mínima de integración es la clase
- Puede ser necesario el uso de drivers o stubs.

Prueba de validación en el contexto OO

Como la validación convencional, la del software OO se enfoca en las acciones y salidas visibles para el usuario.

Para armar los casos de prueba se recurre a casos de uso del modelo de requerimientos.

Los métodos convencionales de prueba de caja negra pueden usarse para hacer pruebas de validación.

Métodos de prueba orientados a objetos

En esta sección:

- Implicaciones de la OO en el diseño de casos de prueba
- Aplicabilidad de métodos convencionales de diseño de casos de prueba
- Casos de prueba y jerarquía de clase
- Diseño de pruebas basadas en escenario

Implicaciones de la OO en el diseño de casos de prueba

Problema	Descripción	Solución
Encapsulamiento	Crea un obstáculo cuando se prueba. Las pruebas requieren reportar el estado de un objeto.	Proporcionar operaciones internas a fin de reportar los valores de los atributos.
Herencia	Si la superclase y la subclase se usan en contextos completamente diferentes, los casos de prueba de superclase tendrán poca aplicabilidad.	Debe diseñarse un nuevo conjunto de pruebas para la subclase.

Aplicabilidad de métodos convencionales de diseño de casos de prueba

Los métodos de caja blanca se pueden usar para probar la implementación de las operaciones.

Los métodos de caja negra son tan aplicables como en los sistemas tradicionales.

Casos de prueba y jerarquía de clase

La herencia no exceptúa de la necesidad de pruebas amplias de todas las clases derivadas.

Si una clase derivada hereda una operación sin cambiarla, aun así la operación podría invocar otras operaciones que sí cambiaron. En ese caso debería volver a probarse.

Diseño de pruebas basadas en escenario

Permiten detectar dos tipos principales de errores: 1) especificaciones incorrectas y 2) interacciones entre subsistemas.

La prueba basada en escenario funciona a un nivel más alto. Se concentra en lo que hace el usuario, no en lo que hace el producto.

Esto significa capturar las tareas (por medio de casos de uso) que el usuario tiene que realizar y luego aplicar éstas y sus variantes como pruebas.

Métodos de prueba aplicables en el nivel clase

En esta sección:

- ❑ Prueba aleatoria para clases OO
- ❑ Prueba de partición en el nivel de clase

Métodos de prueba aplicables en el nivel clase

La prueba “en lo pequeño” se enfoca en una sola clase y en los métodos que encapsula ésta.

Para probar una clase, puede usarse la prueba aleatoria, pero lleva a muchas combinaciones.

Para mejorar la eficiencia de la prueba, puede usarse una estrategia similar a la partición de equivalencia.

Prueba aleatoria para clases OO

Esta técnica consiste en seleccionar al azar diferentes casos de prueba válidos para la vida de un objeto.

Prueba de partición en el nivel de clase

Para reducir el número de casos de prueba requeridos para verificar una clase, las entradas y salidas se categorizan y los casos diseñan para probar cada categoría.

¿Pero cómo se derivan las categorías de partición?

Prueba de partición en el nivel de clase (cont.)

La respuesta puede ser:

- Partición basada en estado
- Partición basada en atributo
- Partición basada en categoría

Partición basada en estado

Categoriza las operaciones de clase a partir de su capacidad para cambiar el estado de la clase.

Por ejemplo para una clase **Cuenta** de un banco:

Cambian el estado	No cambian el estado
deposito extraccion	saldo resumen tipoCuenta limiteCredito

Partición basada en categorías

Clasifica las operaciones de clase con base en la función genérica que cada una realiza.

Por ejemplo, para la clase **Cuenta**, podrían clasificarse las operaciones como:

inicialización	cálculo	consulta	terminación
abrir	deposito extraccion	saldo resumen tipoCuenta limiteCredito	cerrar

Diseño de casos de prueba interclase

En esta sección:

- Prueba de clase múltiple

Prueba de clase múltiple

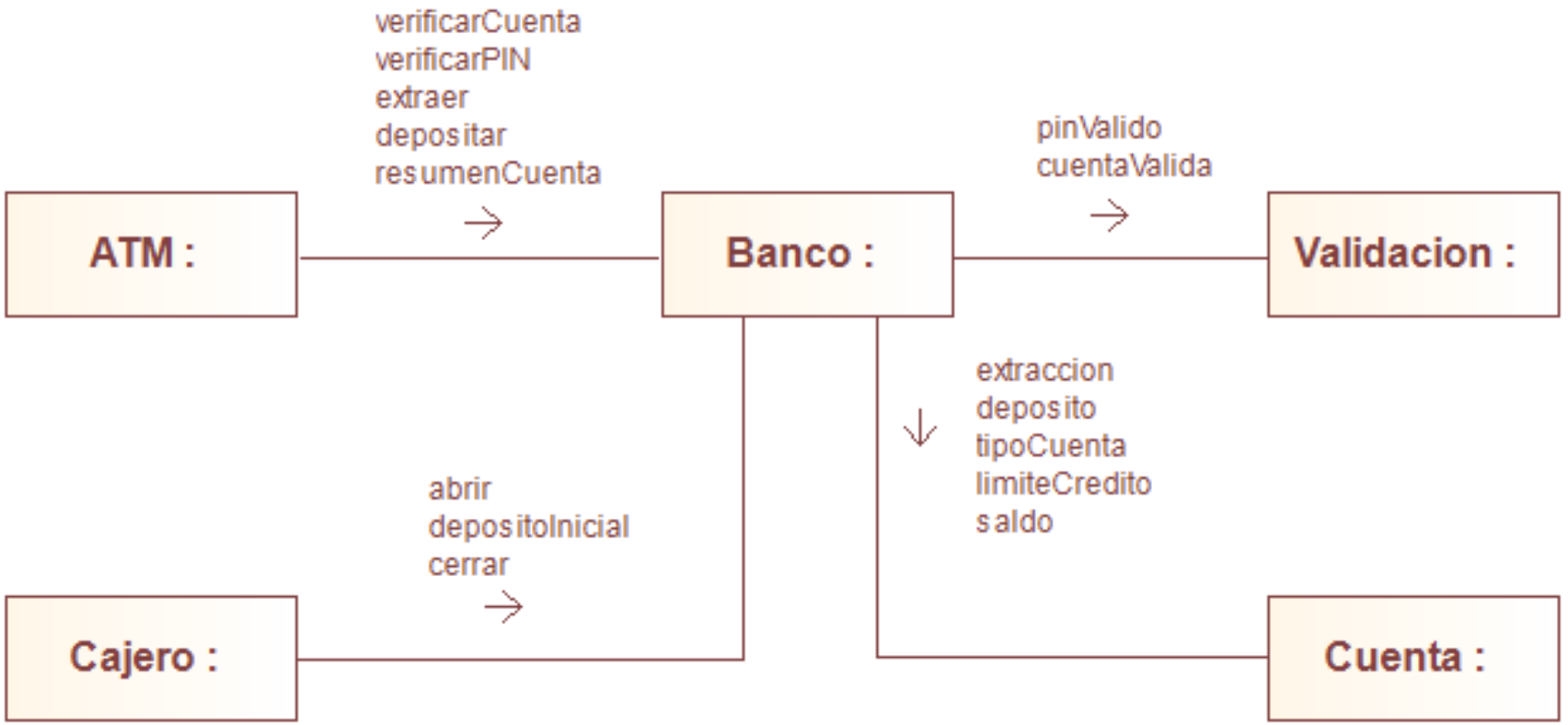
Pasos para generar casos de prueba aleatorios:

1. Para cada clase cliente, use la lista de operaciones para generar secuencias de prueba aleatorias. Las operaciones enviarán mensajes a otras clases servidor.
2. Para cada mensaje generado, determine la clase colaboradora y la correspondiente operación en el objeto servidor.
3. Para cada operación en el objeto servidor, determine los mensajes que transmite.
4. Para cada uno de los mensajes, determine el siguiente nivel de operaciones que se invocan e incorpore esto en la secuencia de prueba.

Prueba de clase múltiple (cont.)

Con respecto a la partición de clase múltiple, una clase individual se particiona como se mostró anteriormente, pero la secuencia de prueba se expande para incluir aquellas operaciones que se invocan mediante mensajes a clases que colaboran.

Prueba de clase múltiple (cont.)



Prueba de clase múltiple (cont.)

Un enfoque alternativo divide las pruebas con base a las interfaces en una clase particular.

En el diagrama, la clase **Banco** recibe mensajes de las clases **ATM** y **Cajero**. Por lo tanto, los métodos dentro de **Banco** pueden probarse al dividirlos en los que sirven a **ATM** y los que sirven a **Cajero**.

La partición basada en estado puede usarse para refinar aún más las particiones.

Esperamos hayan disfrutado y aprovechado del estudio y las actividades propuestas en esta unidad!!!!!!!!!!

UTN.BA FACULTAD REGIONAL BUENOS AIRES

SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

www.sceu.frba.utn.edu.ar/e-learning