OO Testing

SI-318 Testing dan Implementasi Sistem

Rosa Ariani Sukamto, ST

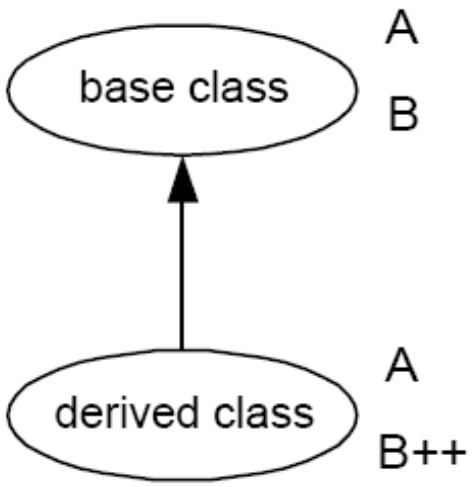
Pengujian Berorientasi Objek

- Tujuan pengujian tetap yaitu untuk menemukan kesalahan dalam selang waktu yang realistik
- Mengubah strategi dan taktik pengujian
- Ada tiga hal yang harus diperhatikan :
  - Definisi pengujian harus diperluas agar mencakup teknik untuk menemukan kesalahan pada model OOA dan OOD
  - Strategi pengujian unit dan integrasi berubah
  - Perancangan pengujian harus memperhatikan karakteristik dari perangkat lunak berorientasi objek

Perancangan Kasus Pengujian (1)

- **Pengaruh OOP pada Pengujian:**
  - muncul beberapa tipe kesalahan yang baru

    Contoh:
    - karena operasi OO biasanya lebih kecil, ada kecenderungan untuk lebih diperlukan adanya integrasi. Dalam hal ini kesalahan integrasi lebih masuk akal.
- **Kasus Pengujian dan Hirarki dari Kelas :**
  - *Inheritance* memerlukan adanya kebutuhan untuk menguji semua kelas turunan
  - memperumit proses pengujian
  - Fungsi B++ harus diuji ulang
  - Fungsi A ?



Perancangan Kasus Pengujian (2)

- Jika fungsi A memanggil fungsi B, dan kelakuan dari B telah berubah, maka A harus diuji ulang walaupun perancangan dan kodenya tidak berubah.
- Jika fungsi A tidak bergantung pada fungsi B maka fungsi A tidak perlu diuji ulang
  - B dan B++
    - dua operasi yang berbeda dengan spesifikasi dan implementasi yang berbeda. Pengujian baru perlu diturunkan hanya untuk kebutuhan fungsi B++ yang tidak dipenuhi oleh pengujian fungsi B.
  - Pengujian untuk B dapat diterapkan pada objek dari kelas B++. Input pengujian mungkin cocok untuk kedua kelas, tapi hasil yang diharapkan dapat berbeda.

Perluasan Sudut Pandang Pengujian (1)

- Kesalahan pendefinisian atribut kelas yang ditemukan pada tahap analisis akan menghilangkan pengaruh yang dapat muncul.
  - Contoh : Sebuah kelas dengan sejumlah atribut didefinisikan pada tahap analisis. Sebuah atribut yang tidak berhubungan dan dua operasi yang memanipulasi atribut tersebut terdefinisi.
  - Jika atribut yang tidak berhubungan dihilangkan pada tahap analisis, dapat mengurangi beberapa masalah dan usaha sbb :
    - Pembuatan subclass yang khusus untuk mengakomodasi atribut tersebut
    - Pembuatan relasi antar kelas yang salah
    - Kelakuan dari sistem dapat menjadi tidak tepat

Perluasan Sudut Pandang Pengujian (2)

- Jika kesalahan tidak ditemukan, masalah yang dapat muncul pada tahap perancangan:
  - penempatan kelas yang tidak tepat pada subsistem
  - perancangan kerja yang tidak perlu
  - model *messaging* (*message connection*) yang tidak tepat
- Jika kesalahan tetap ada sampai pada tahap pengkodean akan menghabiskan banyak waktu dan usaha untuk
  - membuat kode dari atribut dan dua operasi yang tidak diperlukan,
  - membuat *message* untuk komunikasi antar objek

Pengujian Model OOA dan OOD (1)

- **Kebenaran dari model OOA dan OOD**
  - Kebenaran dari sintaks :
    - Penggunaan simbol dan aturan pemodelan yang tepat
  - Kebenaran dari sematik
    - Model yang mewakili dunia nyata, dibutuhkan seorang ahli dalam domain persoalan.
    - Hubungan antar kelas
- **Kekonsistenan dari model OOA dan OOD**
  - hubungan antar entitas dalam model
  - dapat digunakan model CRC dan *object-relationship diagram*

Pengujian Model OOA dan OOD (2)

Class Name : credit sale	
Class Type : transaction event	
Class Characteristics :	
Responsibilities :	**Collaborators :**
read credit card	credit card
get authorization	credit authority

CRC Index card

Pengujian Model OOA dan OOD (3)

- Langkah :
  - Lakukan pemeriksaan silang antara model CRC dengan model *object-relationship* untuk memastikan semua kolaborasi yang dinyatakan dalam OOA direfleksikan dengan tepat dalam kedua model
  - Periksa deskripsi dari setiap CRC index card untuk menentukan apakah suatu tanggung jawab merupakan bagian dari definisi *collaborator*
  - Periksa hubungan balik untuk memastikan bahwa setiap *collaborator* menerima permintaan dari sumber yang tepat.
  - Periksa hubungan balik untuk memastikan apakah kelas lain diperlukan sebagai *collaborator*
  - Tentukan apakah beberapa tanggung jawab dapat digabungkan menjadi tanggung jawab
  - Ke lima langkah di atas diterapkan untuk setiap kelas dan setiap evolusi dari model OOA

Strategi Pengujian (1)

- Strategi : pengujian semua unit program terkecil, pengujian integritas dari modul, dan pengujian keseluruhan sistem
- **Pengujian Unit dalam konteks berorientasi objek**
  - Unit terkecil Kelas atau objek
  - Setiap operasi yang diturunkan pada kelas turunan harus diperiksa
- **Pengujian Integritas dalam konteks berorientasi objek**
  - *Thread-based testing*
    - mengintegrasikan sekumpulan kelas suatu input atau kejadian dalam sistem.
    - Setiap *thread* diintegrasikan dan diuji secara individual.
    - Pengujian regresi diterapkan untuk memastikan tidak ada efek samping yang muncul.

Strategi Pengujian (1)

- *Use-based testing*
  - Pengujian terhadap setiap *independent classes*
  - Pengujian terhadap *dependent classes* sampai keseluruhan sistem terbentuk
- *Cluster testing* : salah satu langkah dalam pengujian integritas, memeriksa kolaborasi antar kelas pada model CRC dan *object-relationship*
- **Pengujian Validasi dalam Konteks Berorientasi Objek**
  - memusatkan pada aksi dari user dan keluaran dari sistem yang dapat dikenali user
  - Case
    - membantu untuk menemukan kesalahan pada kebutuhan interaksi user
  - Black Box
    - mengatur pengujian validasi

Metode Pengujian Pada Level Kelas (1)

- **Random Testing**
  - Contoh : Aplikasi perbankan mempunyai kelas **account** yang dengan operasi : *open*, *setup*, *deposit*, *withdraw balance*, *summarize*, *creditLimit*, dan *close*.
    - Sejarah hidup minimum dari **account** adalah operasi *open, setup, deposit, withdraw, close*
    - Variasi yang mungkin muncul
    - Kasus uji lain yang mungkin
- **Partition Testing**
  - mengurangi jumlah kasus uji dengan mengelompokkan input dan output, kasus uji dirancang untuk memeriksa setiap kelompok
  - Ada beberapa cara :
    - **Pembagian berdasarkan status (*state-based partitioning*)**
      - mengelompokkan operasi berdasarkan kemampuan untuk mengubah status dari kelas
      - Contoh pada kelas **account** :
        - operasi yang mengubah status adalah *deposit* dan *withdraw*
        - operasi yang tidak mengubah status adalah *balance*, *summarize*, dan *creditLimit*

Metode Pengujian Pada Level Kelas (2)

- **Pembagian berdasarkan atribut (*attribute-based partitioning*)**
  - mengelompokkan operasi berdasarkan atribut yang digunakan
  - Contoh :
    - operasi yang menggunakan *creditLimit*,
    - operasi yang mengubah *creditLimit*
    - operasi yang tidak menggunakan atau mengubah *creditLimit*.
- **Pembagian berdasarkan kategori (*category-based partitioning*)**
  - mengelompokkan operasi berdasarkan fungsi generik dari setiap operasi
  - Contoh :
    - operasi inisialisasi (*open, setup*)
    - operasi perhitungan (*deposit*, *withdraw*)
    - operasi query (*balance*, *summarize*, *creditLimit*)
    - operasi terminasi (*close*)

Perancangan Kasus Pengujian Antar Kelas (1)

- dimulai pada saat pengintegrasian sistem OO
- dapat dilakukan dengan menerapkan metoda acak dan partisi, pengujian *scenario-based* dan *behavioral*
- **Pengujian Multiple Class**
  - Untuk setiap kelas *client*, gunakan daftar operator untuk membuat sederetan pengujian acak.
  - Untuk setiap pesan yang dibangkitkan, tentukan kelas kolaborator dan operator yang berhubungan pada objek *server*.
  - Untuk setiap operator pada objek *server*, tentukan pesan yang dikirimkan
  - Untuk setiap pesan, tentukan operator level berikutnya yang dibangkitkan dan masukkan ke dalam urutan pengujian.

Perancangan Kasus Pengujian Antar Kelas (2)

- **Pengujian yang Diperoleh dari Behavior Model**
  - *State-transition diagram* (STD) menyatakan kelakukan dinamik dari kelas .
  - STD dapat digunakan untuk membantu dalam mendapatkan urutan pengujian yang akan memeriksa kelakukan dinamik dari kelas dan kelas-kelas yang berkolaborasi dengannya
  - Untuk situasi di mana kelakukan kelas menghasilkan suatu kolaborasi dengan satu kelas atau lebih, gunakan *multiple* STD untuk menelusuri aliran kelakuan dari sistem.
  - Penelusuran status dapat dilakukan dengan menggunakan metoda *breadth first*
    - kasus uji memeriksa sebuah transisi tunggal dan ketika transisi baru diuji, hanya transisi yang sudah diuji yang boleh digunakan.