Testing Object-Oriented Applications

OO Testing

- Untuk menguji sistem OO secara memadai, tiga hal harus dilakukan:
  - definisi pengujian harus diperluas untuk mencakup teknik penemuan kesalahan yang diterapkan pada analisis berorientasi objek dan model desain
  - strategi untuk pengujian unit dan integrasi harus berubah secara signifikan, dan
  - desain kasus uji harus memperhitungkan karakteristik unik dari perangkat lunak OO.

‘Testing’ OO Models

- Tinjauan analisis OO dan model desain sangat berguna karena konstruk semantik yang sama (mis., kelas, atribut, operasi, pesan) muncul pada tingkat analisis, desain, dan kode
- Oleh karena itu, masalah dalam definisi atribut kelas yang ditemukan selama analisis akan menghindari dampak samping yang mungkin terjadi jika masalah tidak ditemukan hingga desain atau kode (atau bahkan iterasi analisis berikutnya).

Correctness of OO Models

- Selama analisis dan desain, kebenaran semantik dapat dinilai berdasarkan kesesuaian model dengan domain masalah dunia nyata.
- Jika model secara akurat mencerminkan dunia nyata (ke tingkat detail yang sesuai dengan tahap pengembangan di mana model ditinjau) maka secara semantik benar.
- Untuk menentukan apakah model benar-benar mencerminkan persyaratan dunia nyata, harus disajikan kepada pakar domain bermasalah yang akan memeriksa definisi kelas dan hierarki untuk kelalaian dan ambiguitas.
- Hubungan kelas (koneksi instan) dievaluasi untuk menentukan apakah mereka secara akurat mencerminkan koneksi objek dunia nyata.

Class Model Consistency

- Tinjau kembali model CRC dan model hubungan-objek.
- Periksa deskripsi masing-masing kartu indeks CRC untuk menentukan apakah tanggung jawab yang didelegasikan adalah bagian dari definisi kolaborator.
- Balikkan koneksi untuk memastikan bahwa setiap kolaborator yang diminta layanan menerima permintaan dari sumber yang masuk akal.
- Dengan menggunakan koneksi terbalik yang diperiksa pada langkah sebelumnya, tentukan apakah kelas lain mungkin diperlukan atau apakah tanggung jawab dikelompokkan dengan benar di antara kelas-kelas tersebut.
- Tentukan apakah tanggung jawab yang diminta secara luas dapat digabungkan menjadi satu tanggung jawab.

OO Testing Strategies

- Unit testing
  - konsep perubahan unit
  - unit terkecil yang dapat diuji adalah kelas enkapsulasi
  - satu operasi tidak dapat lagi diuji secara terpisah (pandangan konvensional dari unit testing) melainkan sebagai bagian dari kelas
- Integration Testing
  - *Thread-based testing* mengintegrasikan sekumpulan kelas yang diperlukan untuk merespons satu input atau peristiwa untuk sistem
  - *Use-based testing* memulai pembangunan sistem dengan menguji kelas-kelas tersebut (disebut kelas independen) yang menggunakan sangat sedikit (jika ada) kelas server. Setelah kelas independen diuji, lapisan kelas berikutnya, disebut kelas dependen
  - *Cluster testing* mendefinisikan sekelompok kelas berkolaborasi (ditentukan dengan memeriksa CRC dan model hubungan-objek) dilakukan dengan merancang kasus uji yang berusaha mengungkap kesalahan dalam kolaborasi.

OO Testing Strategies

- Validation Testing
  - detail koneksi kelas menghilang
  - menggambarkan kasus penggunaan yang merupakan bagian dari model persyaratan
  - Metode pengujian kotak hitam konvensional dapat digunakan untuk mendorong pengujian validasi

OOT Methods

Berard mengusulkan pendekatan berikut:

1. Setiap kasus uji harus diidentifikasi secara unik dan harus secara eksplisit dikaitkan dengan kelas yang akan diuji,
2. Tujuan tes harus dinyatakan,
3. Daftar langkah-langkah pengujian harus dikembangkan untuk setiap tes dan harus berisi:
   a. daftar state yang ditentukan untuk objek yang akan diuji
   b. daftar pesan dan operasi yang akan dilakukan sebagai konsekuensi dari pengujian
   c. daftar pengecualian yang dapat terjadi saat objek diuji
   d. daftar kondisi eksternal (mis., perubahan lingkungan di luar perangkat lunak yang harus ada untuk melakukan pengujian dengan benar)
   e. informasi tambahan yang akan membantu dalam memahami atau melaksanakan tes.

Testing Methods

- Fault-based testing
  - Penguji mencari kesalahan yang masuk akal (mis., aspek implementasi sistem yang dapat mengakibatkan cacat). Untuk menentukan apakah ada kesalahan ini, uji kasus dirancang untuk menggunakan desain atau kode.
- Class Testing and the Class Hierarchy
  - Warisan (*Inheritance*) tidak meniadakan kebutuhan untuk pengujian menyeluruh dari semua kelas turunan. Padahal, hal itu justru bisa menyulitkan proses pengujian
- Scenario-Based Test Design
  - Pengujian berbasis skenario berkonsentrasi pada apa yang dilakukan pengguna, bukan pada apa yang dilakukan produk. Ini berarti menangkap tugas (melalui kasus penggunaan) yang harus dilakukan pengguna, kemudian menerapkannya dan variannya sebagai tes.

OOT Methods: Random Testing

- Random testing
  - mengidentifikasi operasi yang berlaku untuk suatu kelas
  - mendefinisikan kendala pada penggunaannya
  - mengidentifikasi urutan pengujian minimum
  - urutan operasi yang mendefinisikan riwayat hidup minimum kelas (objek)
  - menghasilkan berbagai urutan uji acak (tetapi valid)
  - latihan sejarah kehidupan contoh kelas lainnya (lebih kompleks)

OOT Methods: Partition Testing

- Partition Testing
  - mengurangi jumlah kasus uji yang diperlukan untuk menguji kelas dengan cara yang sama seperti partisi kesetaraan untuk perangkat lunak konvensional
  - partisi berbasis state
    - mengkategorikan dan menguji operasi berdasarkan kemampuan mereka untuk mengubah keadaan kelas
  - partisi berbasis atribut
    - kategorikan dan uji operasi berdasarkan atribut yang mereka gunakan
  - partisi berbasis kategori
    - kategorikan dan uji operasi berdasarkan fungsi generik masing-masing melakukan

OOT Methods: Inter-Class Testing

- Inter-class testing
  - Untuk setiap kelas klien, gunakan daftar operator kelas untuk menghasilkan serangkaian urutan pengujian acak. Operator akan mengirim pesan ke kelas server lain.
  - Untuk setiap pesan yang dihasilkan, tentukan kelas kolaborator dan operator terkait di objek server.
  - Untuk setiap operator di objek server (yang telah dipanggil oleh pesan yang dikirim dari objek klien), tentukan pesan yang ditransmisikan.
  - Untuk setiap pesan, tentukan level operator selanjutnya yang dipanggil dan gabungkan ini ke dalam urutan pengujian

OOT Methods: Behavior Testing

Tes yang akan dirancang harus mencapai semua cakupan state. Artinya, urutan operasi harus menyebabkan kelas Akun melakukan transisi melalui semua state yang diizinkan