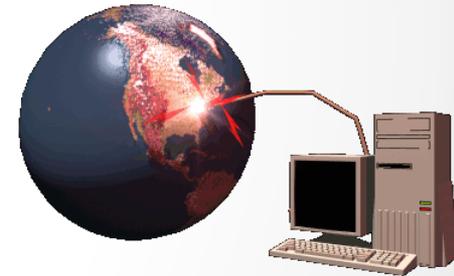




Teknik Informatika S1

SOFTWARE QUALITY AND TESTING

OO Testing



Disusun Oleh:

Egia Rosi Subhiyakto, M.Kom, M.CS

Teknik Informatika UDINUS

egia@dsn.dinus.ac.id

+6285640392988

SILABUS MATA KULIAH

8. Perencanaan Pengujian + Tugas Black Box
9. Presentasi Tugas Black Box
10. White Box (1)
11. White Box (2) + Post Test
12. Manajemen Fungsi Testing
- 13.00 Testing**
14. Tool support for testing + Tugas Pengganti Absen

Pengujian Berorientasi Objek

- Tujuan pengujian tetap yaitu untuk menemukan kesalahan dalam selang waktu yang realistik.
- Mengubah strategi dan taktik pengujian

Pengujian Berorientasi Objek

- 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

PERLUASAN SUDUT PANDANG PENGUJIAN

- 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

PERLUASAN SUDUT PANDANG PENGUJIAN (2)

- 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

■ Kebenaran dari model OOA dan OOD

○ Kebenaran dari sintaks :

➤ Penggunaan simbol dan aturan pemodelan yang tepat

○ Kebenaran dari semantik

➤ Model yang mewakili dunia nyata, dibutuhkan seorang ahli dalam domain persoalan.

➤ Hubungan antar kelas

PENGUJIAN MODEL OOA dan OOD

- **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

STRATEGI PENGUJIAN

- 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

STRATEGI PENGUJIAN

▪ 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 (2)

✓ *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*

STRATEGI PENGUJIAN (2)

▪ **Pengujian Validasi dalam Konteks Berorientasi Objek**

- ✓ memusatkan pada aksi dari user dan keluaran dari sistem yang dapat dikenali user
- ✓ Use Case → membantu untuk menemukan kesalahan pada kebutuhan interaksi user
- ✓ Black Box → mengatur pengujian validasi

PERANCANGAN KASUS PENGUJIAN

▪ Menurut Berard [BER93] :

- ✓ Setiap kasus uji diidentifikasi secara unik dan diasosiasikan secara eksplisit dengan kelas yang akan diuji.
- ✓ Tujuan pengujian harus dinyatakan
- ✓ Daftar tahapan pengujian harus dibuat untuk setiap pengujian:
 - daftar keadaan dari objek yang diuji
 - daftar pesan dan operasi yang muncul
 - daftar exception yang mungkin muncul saat objek diuji
 - daftar kondisi eksternal
 - informasi tambahan yang membantu dalam memahami
 - pengujian atau mengimplementasikan pengujian.

PERANCANGAN KASUS PENGUJIAN (2)

■ **Implikasi Kosep Berorientasi Objek :**

- ✓ Enkapsulasi menyebabkan informasi dari status objek yang sedang diuji sulit diperoleh
- ✓ Inheritance menyebabkan perlu dilakukannya pengujian setiap reuse (jika subclass digunakan dalam konteks yang berbeda dengan superclass-nya)
- ✓ Multi inheritance menyebabkan penambahan konteks yang harus diuji

PERANCANGAN KASUS PENGUJIAN (2)

- ***Applicability* Metoda Perancangan Kasus Uji Konvensional**
 - ✓ White Box → pengujian operasi
 - *basis path, loop testing, atau data flow* → untuk memastikan bahwa setiap pernyataan dalam operasi telah diuji
 - ✓ Black Box → untuk menguji sistem
 - Use case → untuk membuat input dalam perancangan *black box* dan pengujian *state-based*

PERANCANGAN KASUS PENGUJIAN (3)

▪ **Pengujian *Fault-Based* :**

- ✓ Untuk merancang pengujian yang mempunyai kemungkinan besar untuk menemukan kesalahan yang masuk akal
- ✓ Perancangan awal perlu pengujian ini dalam tahap Analisis
- ✓ Efektivitas pengujian tergantung dari cara pandang penguji dalam melihat kesalahan
- ✓ Pengujian integritas → mencari kesalahan yang masuk akal dalam *message connection*. Ada 3 jenis kesalahan yang dapat muncul dalam konteks ini :
 - hasil yang tidak diharapkan
 - kesalahan penggunaan message/ operasi
 - pemanggilan yang salah

PERANCANGAN KASUS PENGUJIAN (4)

■ Pengaruh OOP pada Pengujian :

- beberapa jenis kesalahan menjadi kurang masuk akal
- beberapa jenis kesalahan menjadi lebih masuk akal
- muncul beberapa tipe kesalahan yang baru

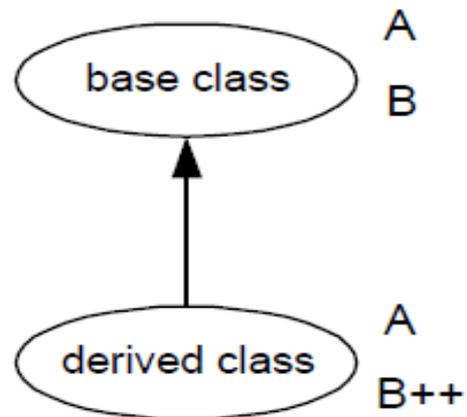
Contoh :

karena operasi OO biasanya lebih kecil, ada kecenderungan untuk lebih diperlukan adanya integrasi. Dalam hal ini kesalahan integrasi lebih masuk akal.

PERANCANGAN KASUS PENGUJIAN (5)

■ 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 (6)

- Jika fungsi A memanggil fungsi B, dan kelakukan 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.

PERANCANGAN KASUS PENGUJIAN (7)

- **Perancangan Pengujian Scenario-Based**
 - Pengujian Fault-based mengabaikan dua kesalahan utama :
 - Kesalahan spesifikasi
 - Kesalahan interaksi antar subsistem
 - Pengujian scenario-based memusatkan pada apa yang dilakukan oleh user, bukan pada apa yang dilakukan produk.
 - Use cases digunakan untuk menentukan apa yang dilakukan oleh user kemudian menerapkannya sebagai pengujian.
 - Pengujian scenario-based cenderung untuk memeriksa banyak subsistem dalam suatu pengujian tunggal.

PERANCANGAN KASUS PENGUJIAN (8)

➤ Contoh perancangan pengujian *scenario-based* untuk text editor:

Use Case : *Fix the Final Draft*

Background : Use case ini menggambarkan urutan *event* yang muncul

1. Cetak seluruh dokumen
2. Periksa dokumen, ubah halaman tertentu
3. Untuk setiap halaman yang diubah, cetak halaman tersebut
4. Kadang-kadang sederetan halaman dicetak

PERANCANGAN KASUS PENGUJIAN (8)

- Kebutuhan user :
 - metoda untuk mencetak satu halaman
 - metoda untuk mencetak beberapa halaman berurutan
- Yang diuji : pengeditan setelah pencetakan
- Penguji berharap untuk menemukan bahwa fungsi pencetakan menyebabkan kesalahan dalam fungsi pengeditan

PERANCANGAN KASUS PENGUJIAN (9)

▪ **Testing Surface Structure dan Deep Structure**

➤ **Surface Structure :**

- struktur yang terlihat secara eksternal dari program berorientasi objek → struktur yang langsung berhubungan dengan end user.
- menggunakan daftar semua objek dari semua objek sebagai checklist pengujian.
- Perancangan kasus uji harus menggunakan objek dan operasinya sebagai petunjuk yang menuntun pada task yang terabaikan

PERANCANGAN KASUS PENGUJIAN (9)

▪ **Testing Surface Structure dan Deep Structure**

➤ **Deep Structure :**

- detail teknis internal dari program berorientasi objek → struktur yang dimengerti dengan memeriksa perancangan dan/atau kode.
- dirancang untuk memeriksa ketergantungan, kelakukan, dan mekanisme komunikasi yang telah dibuat

METODA PENGUJIAN PADA LEVEL KELAS

▪ **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 :

*open • setup • deposit • [deposit / withdraw / balance
/ summarize / creditLimit] n • withdraw • close*

Kasus uji lain yang mungkin:

*open • setup • deposit • deposit • balance • summarize
• withdraw • close*

METODA PENGUJIAN PADA LEVEL KELAS(2)

■ Partition Testing

- mengurangi jumlah kasus uji dengan mengelompokkan input dan output, kasus uji dirancang untuk memeriksa setiap kelompok.
- Ada beberapa cara :

1. 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*

METODA PENGUJIAN PADA LEVEL KELAS(2)

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*.

METODA PENGUJIAN PADA LEVEL KELAS(2)

3. 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*)
- dimulai pada saat pengintegrasian sistem OO
- dapat dilakukan dengan menerapkan metoda acak dan partisi, pengujian *scenario-based* dan *behavioral*

Perancangan Kasus Pengujian Antar Kelas

▪ Pengujian Multiple Class

Kirani dan Tsai [KIR94] menyarankan tahapan sebagai berikut:

1. Untuk setiap kelas *client*, gunakan daftar operator untuk membuat sederetan pengujian acak.
2. Untuk setiap pesan yang dibangkitkan, tentukan kelas kolaborator dan operator yang berhubungan pada objek *server*.
3. Untuk setiap operator pada objek *server*, tentukan pesan yang dikirimkan
4. 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 kelakuan dinamik dari kelas.
- STD dapat digunakan untuk membantu dalam mendapatkan urutan pengujian yang akan memeriksa kelakuan dinamik dari kelas dan kelas-kelas yang berkolaborasi dengannya.

Perancangan Kasus Pengujian Antar Kelas (2)

■ **Pengujian yang Diperoleh dari Behavior Model**

- Untuk situasi di mana kelakuan 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.

TERIMA KASIH

Tugas untuk pengganti absen

Kerjakan Tugas pengganti absen yang saya upload di Siadin (Tugas Pengganti Absen.pdf)

Ikuti instruksi berikut:

- ✓ Buat kelompok masing-masing 3 orang/ kelompok
- ✓ Kombinasi tugas yang dikerjakan (Akan diacak hari ini):
 1. Tes1, Tes3, Tes5, Tes7, Tes12
 2. Tes2C, Tes2D, Tes4, Tes6, Tes27
 3. Tes4, Tes5, Tes11, Tes27
 4. Test12, Tes27, Tes19
 5. Algo106
- ✓ Kerjakan tugas (diketik dan diprint tanpa jilid) dengan menggunakan **Basis Path Testing** (Step 1-4)
- ✓ Kumpulkan antara tanggal 22-30 Desember 2014 + Absen terakhir



SQT 4804

1. Dedy → 3
2. Heri → 4
3. Peter → 3
4. Setiyo → 2
5. Huda → 2
6. Monica → 5
7. Munawir → 5



Note: Bagi yang belum dapat kelompok silahkan buat kelompok sendiri, dan kerjakan No 5

SQT 4803

1. David → 2
2. Agus F → 2
3. Adhi → 5
4. Aan → 3
5. Martita → 1
6. Septiadi → 4
7. Eko Adi → 2
8. Deni N → 1
9. Pandu → 1
10. Haddawi → 5
11. Kevin → 4
12. Setya → 4



Note: Bagi yang belum dapat kelompok silahkan buat kelompok sendiri, dan kerjakan No 5