Testing Fungsional Sistem

Arna Fariza

Materi

- Tujuan Testing
- Test kasus, test kesesuaian dan test data
- Apakah testing fungsional sistem itu?
- Ulasan
- Teknik testing fungsional sistem:
  - Analisa fungsional
  - Pembagian kesesuaian
  - Analisa batasan nilai

Tujuan Testing Software

- Proses pembuktian nyata dari kerusakan sistem software
  - Tidak termasuk usaha yang berhubungan dengan pelacakan bug dan perbaikannya
- Tanpa batasan jumlah testing akan meningkatkan kualitas program komputer
  - Semakin banyak testing yang dilakukan pada sistem, semakin meyakinkan kebenarannya\
  - Testing tidak dapat membuktikan sistem bekerja dengan benar 100%

Test case

- Komponen dasar dari testing adalah **Test Case**
- Dalam bentuk yang paling umum: (*input, hasil-yg-diharapkan)*
  - *Input* terdiri dari state sistem, perintah user dan nilai data yang diproses
  - *Hasil yang diharapkan* terdiri dari perubahan antar muka yang tampak/terdengar atau perubahan state sistem
- Test case diorganisasi dalam **Test Suite**
  - Fungsionalitas, keamanan, kehandalan, …

Eksekusi Test case

- Menjalankan software (pada waktu test) yang menyediakan input tertentu dalam test case dan mengamati hasil dan membandingkannya sesuai yang ditentukan dalam test case
  - Jika hasil aktual bervariasi dari hasil yang diharapkan, maka kegagalan dapat terdeteksi

Data Test

- Strategi test yang efektif membutuhkan akuisisi dan persiapan yang hati-hati terhadap data test sebelum testing
  - Testing akan sulit jika data test kurang baik
- Data test menyangkut:
  - *Kedalaman*: kuantitas dan ukuran data
  - *Jenis data*: varian nilai data dan tipe data
  - *Bidang*: kelengkapan, relevansi dan akurasi data
    - Hasil query seharusnya valid untuk query tertentu dan tidak seharusnya menjadi nilai yang hilang atau tidak tepat
  - *Kondisi*: data seharusnya menyatakan "kondisi" tertentu dalam domain
    - Data tidak seharusnya setelah membentuk operasi tertentu
- Data test dan hasil test mahal untuk dibangun

Contoh: Data Test untuk TVRS

Nama: test1.db			
Deskripsi: Pelanggaran menyimpan validating violation lookup			
Violation ID	Offender's first name	Offender's last name	Issuing policeman ID
243567	Rachel	Josef	8700342
237812	Dan	Levi	6386541
264683	Dan	Porat	1346329
255245	Dina	Josef	8245731
000345	longFirstN	longLastN	8700342
…	…	…	…

Spesifikasi vs. implementasi

- Pendekatan dasar untuk testing software berdasarkan *spesifikasi* dan *implementasi*
- **White box testing** – test case dan data dibangun berdasarkan kode yang mengimplementasikan software
  - Kualitas dan kebenaran komputasi divalidasi
- **Black box testing** – test case dan data dibentuk semata-mata pada spesifikasi software
- **Gray box testing** – test case dibentuk yang langsung menarget modul dan layer sistem (membutuhkan pengetahuan arsitektur)

Testing Fungsional Sistem

- Testing aplikasi lengkap untuk menyatakan semua perilaku terpenuhi
  - Testing keseluruhan increment yang merupakan tingkat fungsional end-user
- Pencarian kerusakan yang berbeda antara operasi aktual sistem dan kebutuhan sistem
- Sistem dianggap sebagai black box / gray box

Berapa banyak testing yang cukup?

- Melengkapi validasi pembagian floating-point pada IEEE 754 membutuhkan test-case sebanyak 2^{64}

```
float divide(float x, float y)
```

- Dari perspektif praktis dan ekonomi, testing yang mendalam biasanya tidak dimungkinkan
  - Bagian software mana yang seharusnya di-tes?
  - Test case apa yang dipilih?

Ulasan

- **Ulasan atau Coverage** adalah ukuran bagaimana kelengkapan suatu test melatih kapabilitas bagian software
  - "setiap baris kode seharusnya dieksekusi setidaknya 1 kali"
  - "satu test case seharusnya dibentuk dari setiap kebutuhan tertentu"
- Perlu menggunakan teknik testing yang memperkecil jumlah test case yang menghasilkan ulasan test yang terluas dengan usaha terendah

Tehnik: Analisa Fungsional

- Menganalisa perilaku yang diharapkan dari sistem berdasarkan spesifikasi fungsional
- Membangkitkan prosedur tes untuk setiap skenario yang mungkin
  - Berhubungan dengan skenario use case
  - Menganalisa bagaimana perubahan satu bagian dari sistem berakibat ke bagian lain
  - Test case "grand tour" : hasil satu tes case menghasilkan data yang diinputkan ke test case berikutnya

Contoh: Analisa Fungsional I

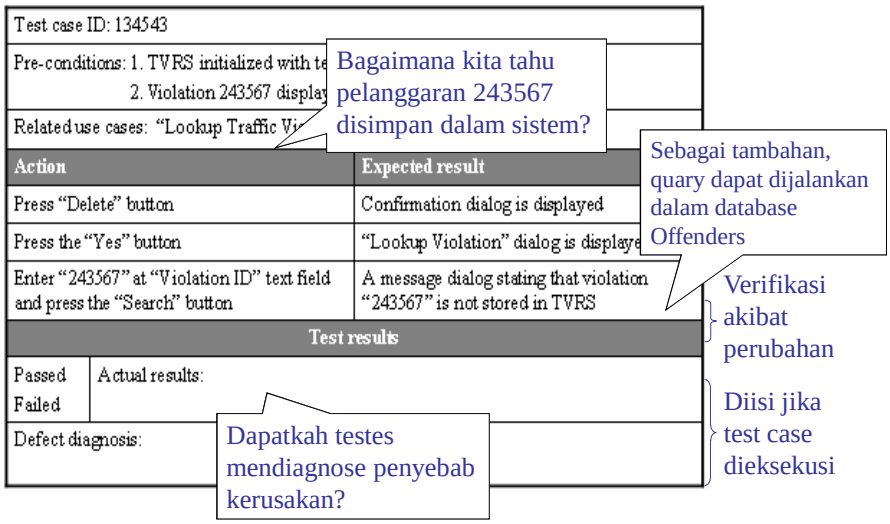
Use Case: Remove Traffic Violation

1. Supervisor memanggil penghapusan Traffic Violation terpilih
2. TVRS mempersilakan supervisor melakukan konfirmasi
3. Supervisor mengkonfirmasi
4. TVRS meminta OffendersDB untuk menghapus Traffic Violation dari data offender
5. OffendersDB menyetujui Traffic Violation dihapus
6. TVRS mengijinkan Supervisor mencari Traffic Violation baru seperti digambarkan dalam use case “Lookup Traffic Violation”

Contoh: Analisa Fungsional II

Test case ID: 134543	
Pre-conditions: 1. TVRS initialized with test1.db database 2. Violation 243567 displayed in the “Lookup Violation” dialog	
Related use cases: “Lookup Traffic Violation”, “Remove Traffic Violation”	
Action	**Expected result**
Press “Delete” button	Confirmation dialog is displayed
Press the “Yes” button	“Lookup Violation” dialog is displayed
Enter “243567” at “Violation ID” text field and press the “Search” button	A message dialog stating that violation “243567” is not stored in TVRS
Test results	
Passed Failed	Actual results:
Defect diagnosis:	

Contoh: Analisa Fungsional III



Tehnik: Partisi yang sama

- Identifikasi jangkauan input dan kondisi inisial yang diharapkan untuk menghasilkan hasil yang sama
- Kelompok test case membentuk class yang sama jika:
  - Melakukan test fitur/skenario yang sama
  - Jika satu test menyatakan gagal, maka test lain (mungkin) akan gagal juga
  - Jika test tidak menghasilkan gagal, maka test yang lain (mungkin) juga tidak akan gagal
- Cukup menggunakan hanya satu class ekuivalen yang representatif

Contoh: Partisi yang Sama I

Spesifikasi nilai input untuk form “Lookup Violation” :

Nama Field	Nilai valid
Violation ID	[0-9]{0, 9}
Offender’s first name	[a-zA-Z]{0, 10}
Offender’s last name	[a-zA-Z]{0, 10}

Contoh: Partisi yang Sama II

Field	Class ekuivalen valid	Nilai representatif valid	Class ekuivalen invalid	Nilai representatif invalid
Violation ID	Known violation	00243567	ID < 0 or ID > 999999999	-1, 1234567890
	Unknown violation	32456720	Non numeric ID	23ab@
	Empty	“”		
Offender’s first name	Unknown violation	David	Character# > 10	Hasalongname
	Single known violation	Rachel	Invalid character	ad0@am
	Many known violations	Dan		
	Empty	“”		
…	…	…	…	

Tehnik: Analisa Batasan Nilai

- Berdasarkan pengalaman/heuristik
  - Testing *kondisi batasan* dari class ekuivalen lebih efektif
- Pilih nilai batas input sebagai representasi class ekuivalen
- Pilih input yang menyatakan batasan nilai output
- Contoh:
  - (0, 10] ⇒ validasi menggunakan 0, 1, 2, 9, 10, 11
  - Baca sampai 5 elemen ⇒ validasi membaca elemen 0, 1, 4, 5, 6