#01
Pengantar Pengujian Perangkat Lunak

PAIK6602 (3 sks)

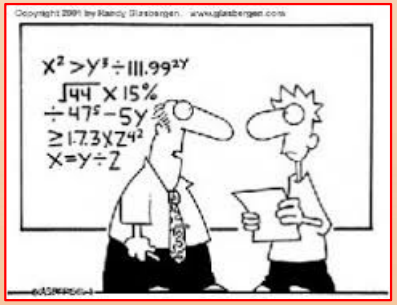
UJI PERANGKAT LUNAK

Nurdin Bahtiar, MT

Pengingat

Jargon

No programming technique solves all problems



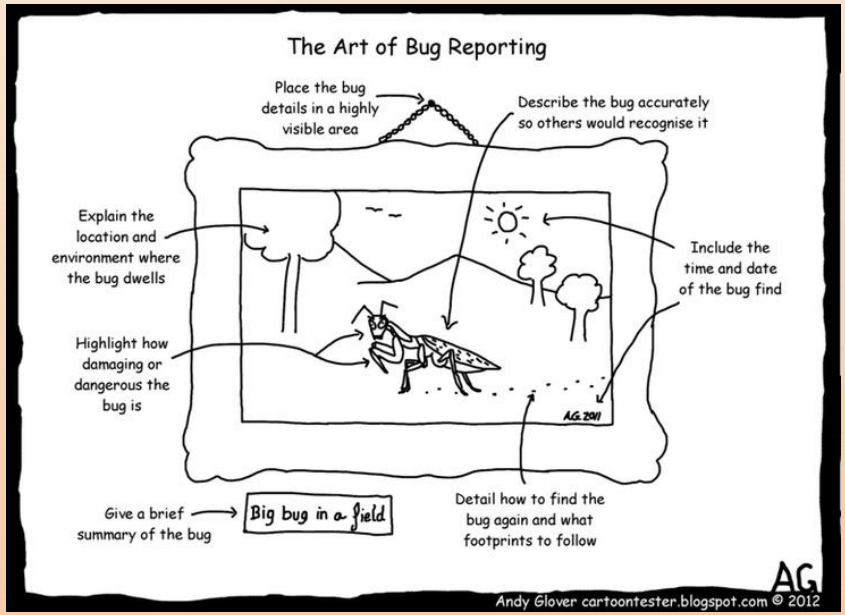
No programming language produces only correct results

No programmer should start each project from scratch

Dasar-dasar Pengujian PL

Pengujian Perangkat Lunak merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari **spesifikasi**, **desain**, dan **pengkodean**.

Dasar-dasar Pengujian PL



Meningkatnya *visibilitas* (kemampuan) perangkat lunak sebagai suatu elemen sistem dan "biaya" yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti.

Pada dasarnya, pengujian merupakan satu langkah dalam proses rekayasa perangkat lunak yang dapat dianggap sebagai hal yang merusak daripada membangun.

Sasaran-sasaran Pengujian

Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.

Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.

Sasaran-sasaran Pengujian

- **Pengujian yang sukses** adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.
- Sasaran tsb berlawanan dengan pandangan yang biasanya dipegang yang menyatakan bahwa pengujian yang berhasil adalah pengujian yang tidak ada kesalahan yang ditemukan.

Prinsip Pengujian

- Semua pengujian harus **dapat ditelusuri** sampai ke persyaratan pelanggan
- Pengujian harus **direncanakan lama** sebelum pengujian itu mulai, maksudnya semua pengujian dapat direncanakan dan dirancang sebelum semua kode dijalankan
- **Prinsip Pareto berlaku** untuk pengujian perangkat lunak, maksudnya dari 80% kesalahan yang ditemukan selama pengujian dapat ditelusuri sampai 20% dari semua modul program
- Pengujian harus **mulai "dari yang kecil"** dan berkembang ke pengujian "yang besar"
- Pengujian yang **mendalam tidak mungkin dilakukan**
- Untuk menjadi paling efektif, pengujian harus **dilakukan oleh pihak ketiga** yang independen.

Testabilitas

- Maksudnya adalah seberapa mudah sebuah program komputer dapat diuji.
- Karena pengujian sangat sulit, perlu diketahui apa yang dapat dilakukan untuk membuatnya menjadi mudah.

- Atribut pengujian yang baik:
  - ✓ Memiliki probabilitas yang tinggi untuk menemukan kesalahan
  - ✓ Tidak redundan
  - ✓ Merupakan “jenis terbaik” (mewakili semua kemungkinan)
  - ✓ Tidak terlalu sederhana atau terlalu kompleks.

Testabilitas

Karakteristik perangkat lunak yang diuji:

- ✓ Operabilitas

  Semakin baik dia bekerja, semakin efisien dia dapat diuji.

- ✓ Observabilitas

  Apa yang Anda lihat adalah apa yang Anda uji.

- ✓ Kontrolabilitas

  Semakin baik kita dapat mengontrol perangkat lunak, semakin banyak pengujian yang dapat diotomatisasi dan dioptimalkan.

- ✓ Dekomposabilitas

  Dengan mengontrol ruang lingkup pengujian, kita dapat dengan lebih cepat mengisolasi masalah dan melakukan pengujian kembali secara lebih luas.

Testabilitas

- ✓ Kesederhanaan

  Semakin sedikit yang diuji, semakin cepat kita dapat mengujinya.

- ✓ Stabilitas

  Semakin sedikit perubahan, semakin sedikit gangguan dalam pengujian.

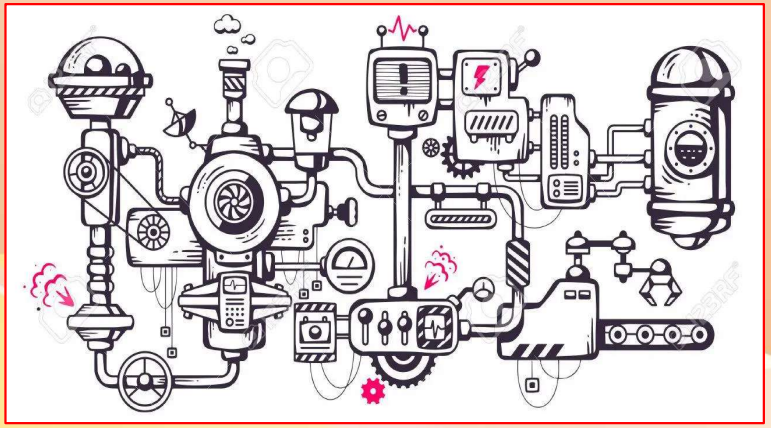
- ✓ Kemampuan untuk dapat dipahami

  Semakin banyak informasi yang kita miliki, semakin halus pengujian yang akan dilakukan.

Desain Test Case

Semua produk yang direkayasa dapat diuji dengan satu atau dua cara:

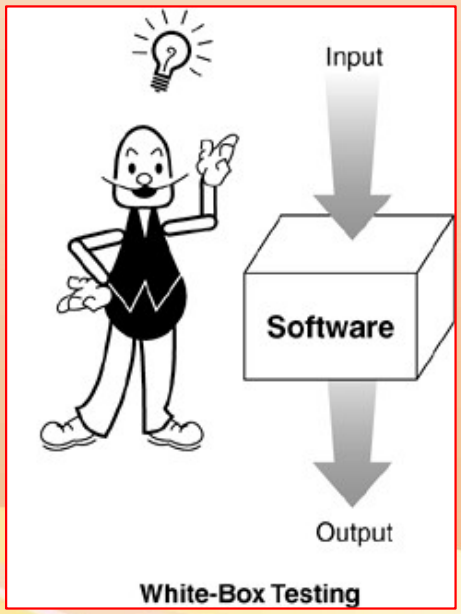
Dengan mengetahui fungsi yang ditentukan di mana produk dirancang untuk melakukannya, pengujian dilakukan untuk memperlihatkan bahwa masing-masing fungsi beroperasi sepenuhnya, pada waktu yang sama mencari kesalahan pada setiap fungsi.



Dengan mengetahui kerja internal suatu produk, maka pengujian dapat dilakukan untuk memastikan bahwa "semua roda gigi berhubungan", yaitu operasi internal bekerja sesuai spesifikasi dan semua komponen internal telah diamati dengan baik.

Pengujian White-Box

- Kadang disebut pengujian **Glass-box**.
- Digunakan untuk memperlihatkan bahwa fungsi-fungsi perangkat lunak adalah operasional; bahwa input diterima dengan baik dan output dihasilkan dengan tepat; dan intergritas informasi eksternal (seperti file data) dapat dipelihara.



- Beberapa karakteristik pengujian white-box:
  1. Menjamin semua jalur independen pada suatu modul digunakan paling tidak satu kali.
  2. Menggunakan semua keputusan logis pada sisi true dan false.
  3. Mengeksekusi semua loop pada batasan mereka dan pada batas operasional mereka.
  4. Menggunakan struktur data internal untuk menjamin validitasnya.

Pengujian White-Box

- Mengapa pengujian white-box saja tidak cukup?

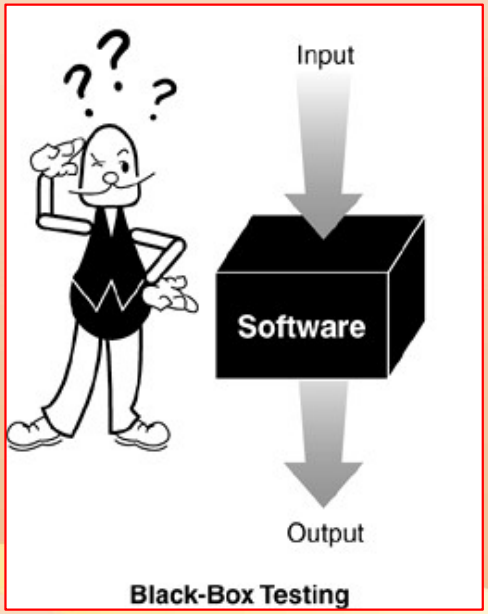
  Karena pengujian mendalam kadang menimbulkan masalah logistik.

- Contoh:
  - ✓ Perhatikan program 100 baris dalam bahasa C. Setelah beberapa deklarasi data, program berisi dua loop tersarang yang masing-masing mengeksekusi 1 sampai 20 kali, tergantung kondisi yang ditentukan pada input.
  - ✓ Pada loop bagian dalam, terdapat empat bangun *if-then-else*. Kira-kira terdapat 10^{14} jalur yang mungkin dapat dieksekusi pada program ini. Berarti dibutuhkan 1370 tahun untuk prosesor test case 1 milidetik.




Pengujian Black-Box

- Pengujian black-box berfokus pada persyaratan fungsional perangkat lunak.
- Pengujian black-box berusaha menemukan kesalahan dalam kategori sebagai berikut:

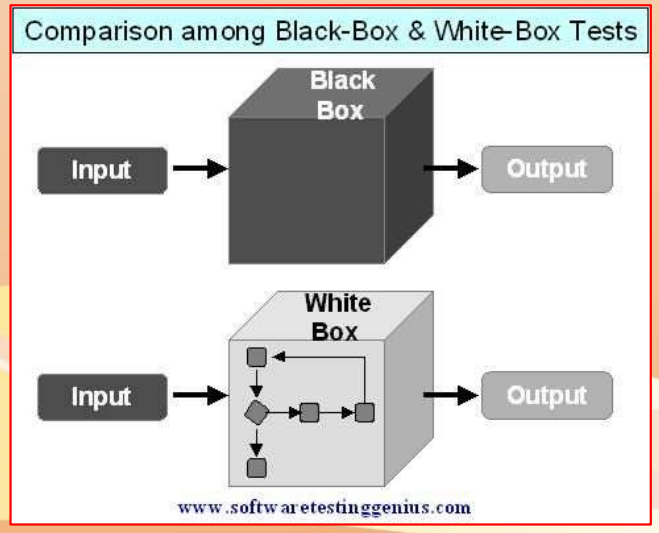


1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses basis data eksternal
4. Kesalahan kinerja
5. Inisiasi dan kesalahan terminasi

Pengujian Black-Box

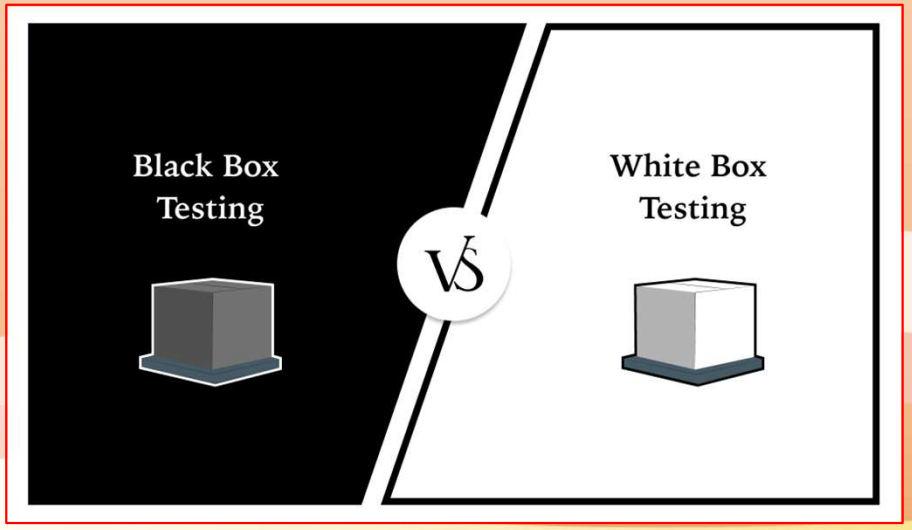
- Mengapa pengujian *black-box* saja tidak cukup?
- Tidak peduli seberapa cermat pengujian black-box dilakukan, dapat tidak menangkap beberapa bentuk kesalahan.
- Beizer (1990):

  "Bug tersembunyi di sudut-sudut dan berkerumun di perbatasan-perbatasan". Tetapi pengujian white-box sangat mungkin untuk melakukannya.



Jenis Pengujian

- Jenis pengujian white-box:
  - ✓ *Basis Path Testing*
  - ✓ *Control Structure Testing*
- Jenis pengujian black-box:
  - ✓ *Graph-based Testing*
  - ✓ *Equivalence Partitioning*
  - ✓ *Boundary Value Analysis*
  - ✓ *Comparison Testing*



End of File

Latihan

1. Pilihlah sebuah jenis produk (dalam kehidupan sehari-hari), lalu tentukan 2 jenis kasus uji serta kualitas yang diharapkan menggunakan:
   - ✓ Pengujian *black-box*
   - ✓ Pengujian *white-box*
2. Sebutkan 2 (dua) saja *requirement* dari program aplikasi yang sedang / pernah Anda kembangkan!

   Pengujian apa yang bisa Anda lakukan pada kedua hal di atas?