SOFTWARE TESTING

Ratna Wardani

Capaian

- Memahami pentingnya *Software Testing*
- Memahami teknik dalam *Software Testing*

Dasar-dasar Software Testing

Teknik-teknik dalam Software Testing

Here we go…

Dasar-dasar *Software Testing*

1. Sasaran Pengujian
2. Prinsip Pengujian
3. Testabilitas

Terminologi

- *Software testing* adalah proses eksekusi suatu program dengan maksud menemukan kesalahan
- Merupakan elemen kritis dari jaminan kualitas *software* dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean.
- *Software testing* menghabiskan upaya 30-40% dari total pekerjaan proyek

SASARAN PENGUJIAN

Sasaran Pengujian

1. Menjalankan program untuk menemukan error.
2. Test case yang bagus adalah yang memiliki kemungkinan terbesar untuk menemukan error yang tersembunyi.
3. Pengujian yang sukses adalah yang berhasil menemukan error yang tersembunyi

Sasaran Pengujian

- Pengujian didesain secara sistematis untuk mencari kelas kesalahan yang berbeda
- Pengujian dilakukan dalam waktu dan usaha minimum
- Pengujian yang sukses adalah yang berhasil menemukan kesalahan dalam software, dan dapat menunjukkan reliabilitas software
- Pengujian tidak dapat memperlihatkan tidak adanya kesalahan

Karakteristik Pengujian

- Testing dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer
- Teknik testing yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya
- *Testing* diadakan oleh *software developer* dan untuk proyek yang besar oleh *group testing* yang *independent*
- *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing*

PRINSIP PENGUJIAN

Prinsip Pengujian

- Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan
- Pengujian harus direncanakan lama sebelum pengujian itu mulai
- 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”

Prinsip Pengujian

- Pengujian yang mendalam tidak mungkin karena tidak mungkin mengeksekusi setiap kombinasi jalur skema pengujian dikarenakan jumlah jalur permutasi untuk program menengah pun sangat besar.
- Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang *independent*
- Sasaran utama desain test case adalah untuk mendapatkan serangkaian pengujian yang memiliki kemungkinan tertinggi di dalam pengungkapan kesalahan pada perangkat lunak

Testabilitas

- Kemudahan *software* dapat diuji.
- Karakteristiknya:
  - Operability: mudah digunakan.
  - Observability: mudah diamati.
  - Controlability: mudah dikendalikan.
  - Decomposability: mudah diuraikan.
  - Simplicity: lingkup kecil, semakin mudah diuji.
  - Stability: jarang berubah.
  - Understandability: mudah dipahami.

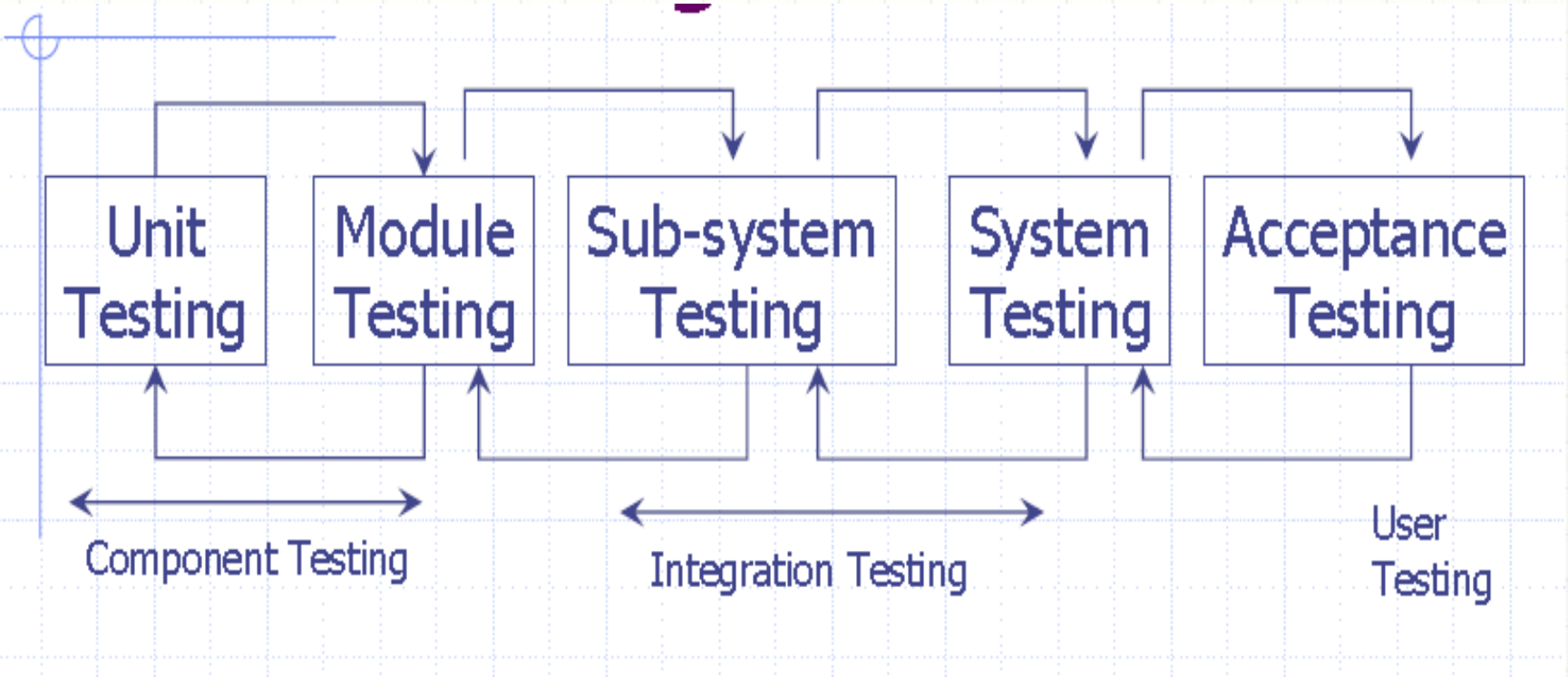
Testabilitas

- Pengujian yang baik memiliki atribut:
  - Memiliki probabilitas yang yang tinggi untuk menemukan kesalahan
  - Tidak redundan. Setiap pengujian harus memiliki tujuan yang berbeda
  - Memiliki probabilitas yang besar menemukan kelas kesalahan yang tinggi
  - Tidak terlalu sederhana dan tidak terlalu kompleks

Teknik-teknik *Software Testing*

1. White Box Testing
2. Black Box Testing
3. Testing dengan Kasus Khusus

Proses Testing



Proses Testing

- Component Testing
  - Pengujian komponen program
  - Dilakukan oleh component developer
- Integration Testing
  - Pengujian kelompok komponen yangterintegrasi membentuk sub-sistem atau sistem
  - Dilakukan oleh tim penguji yang independen
  - Pengujian berdasar spesifikasi sistem
- User Testing
  - Pengujian fungsionalitas sistem

White Box Testing

Lingkup Pengujian

- Pengujian dilakukan untuk mengetahui kerja internal produk/sistem yang dihasilkan
  - Pengamatan detail prosedur.
  - Mengamati sampai level percabangan kondisi dan perulangan.
- Metode
  - Pengujian Basis Path
  - Pengujian Struktur Kontrol

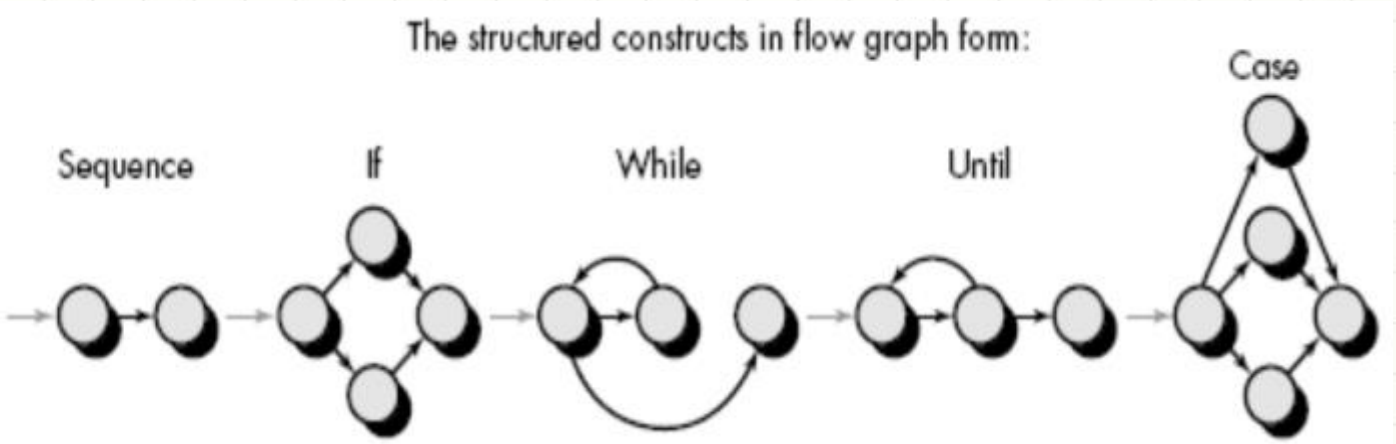
Pengujian Basis Path

- Menggunakan Notasi Diagram Alir
- Jenis
  - Kompleksitas Siklomatis
  - Test Case
  - Matriks Grafik

Pengujian Basis Path

- Notasi diagram alir
  - Menggambarkan aliran kontrol logika
  - Setiap representasi desai prosedural dapat digambarkan dengan satu diagram alir

FIGURE 17.1
Flow graph notation



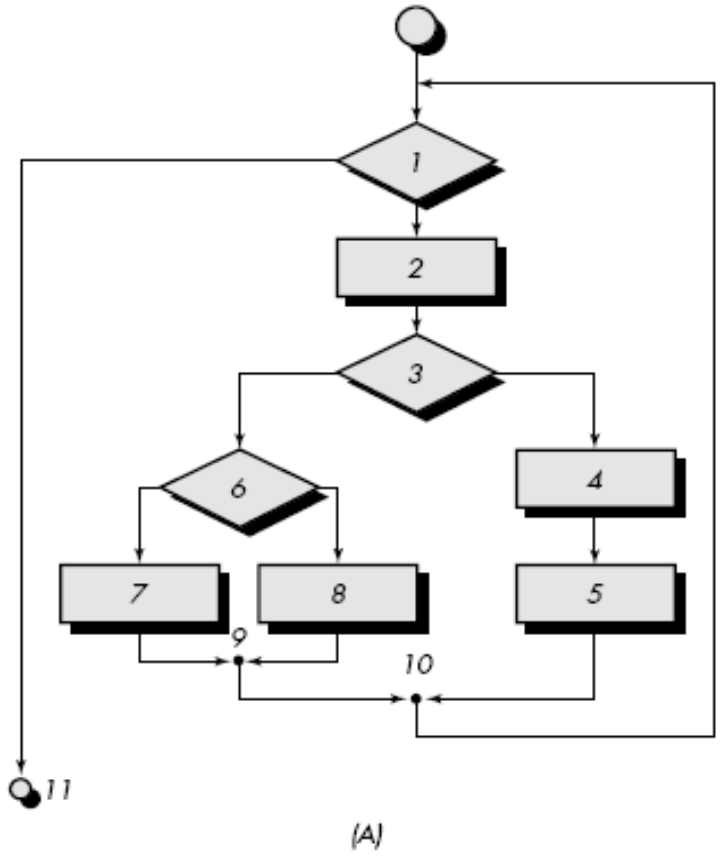
Pengujian Basis Path

- Kompleksitas Siklomatis
  - Menunjukkan jumlah skenario pengujian yang harus dilakukan untuk menjamin cakupan seluruh program.
  - Dasar : teori Graf
  - V(G) = E-N+2

    V(G) : kompleksitas siklomatis

    E : jumlah edge

    N : jumlah simpul



Pengujian Basis Path

- Test Case
  - Berdasar desain atau kode sumber, gambarkan grafik alir yang sesuai
  - Tentukan kompleksitas siklomatisdari grafik alir
  - Tentukan sebuah basis path dari jalur independen secara linier → harga V(G)
  - Siapkan test case yang akan memaksa adanya eksekusi setiap basis set

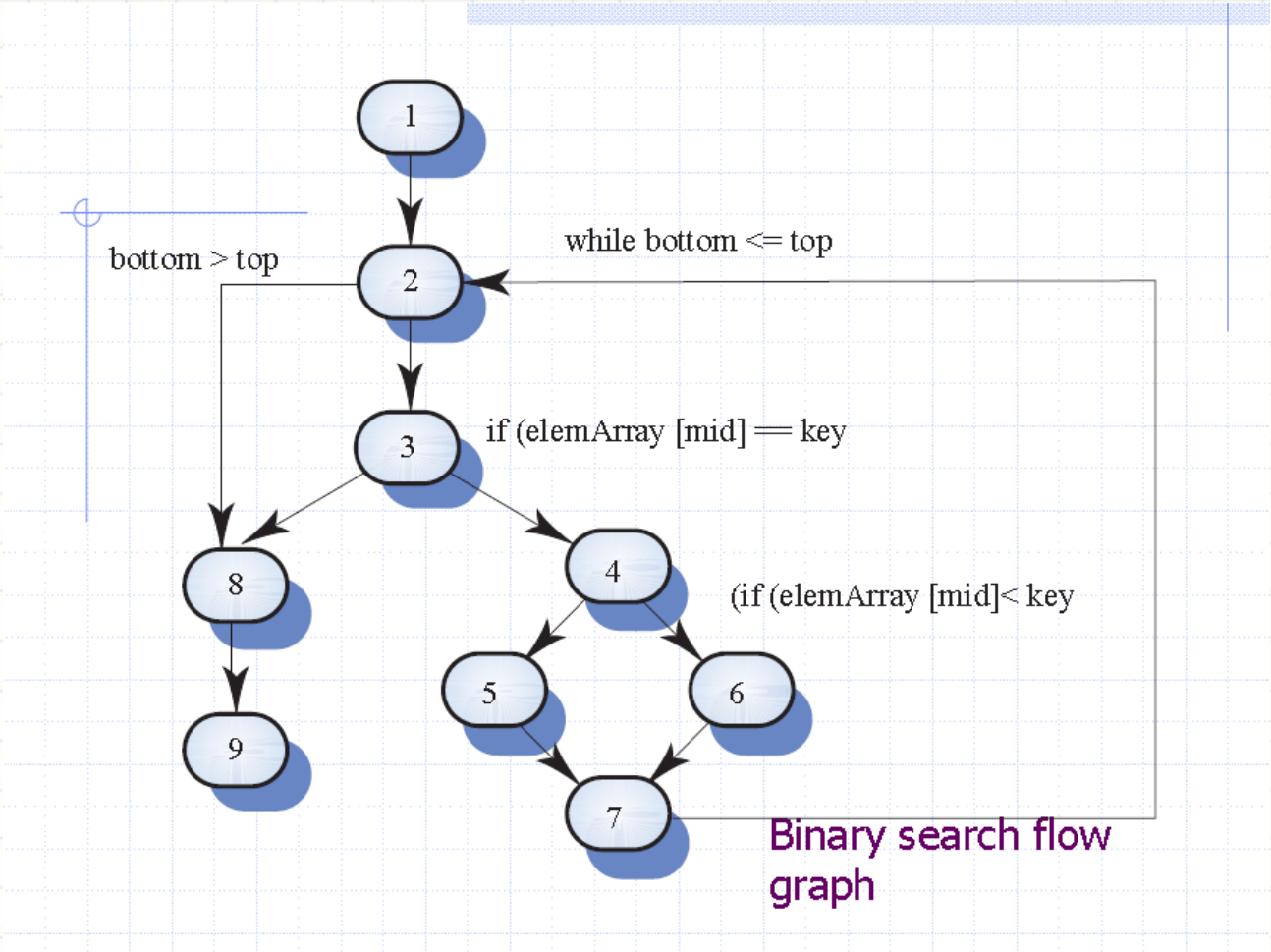
Pengujian Basis Path

- Test Case

```
PROCEDURE average;

    *  This procedure computes the average of 100 or fewer
       numbers that lie between bounding values; it also computes the
       sum and the total number valid.

    INTERFACE RETURNS average, total.input, total.valid;
    INTERFACE ACCEPTS value, minimum, maximum;

    TYPE value[1:100] IS SCALAR ARRAY;
    TYPE average, total.input, total.valid;
       minimum, maximum, sum IS SCALAR;
    TYPE i IS INTEGER;
1   i = 1;
    total.input = total.valid = 0;                    2
    sum = 0;
    DO WHILE value[i] <> -999 AND total.input < 100   9
4      increment total.input by 1;
       IF value[i] >= minimum AND value[i] <= maximum 6
5         THEN increment total.valid by 1;
7                 sum = sum + value[i]
          ELSE skip
8      ENDIF
       increment i by 1;
9   ENDDO
    IF total.valid > 0                                10
11     THEN average = sum / total.valid;
12        ELSE average = -999;
10  ENDIF
END average
```

Pengujian Basis Path

- Test Case



Pengujian Basis Path

- Test Case
  - 1, 2, 3, 8, 9
  - 1, 2, 3, 4, 6, 7, 2
  - 1, 2, 3, 4, 5, 7, 2
  - 1, 2, 3, 4, 6, 7, 2, 8, 9
  - Test cases harus ditentukan sehingga semua path tsb tereksekusi.

Pengujian Basis Path

- Matriks Grafik
  - Digunakan untuk mengevaluasi kontrol program selama pengujian
  - Matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir
  - Masing-masing baris dan kolom sesuai dengan simpul yang diidentifikasi dan entri matriks sesuai dengan edge di antara simpul

Pengujian Struktur Kontrol

- Metode desain test case yang menggunakan kondisi logis / struktur program
- Jenis:
  - Pengujian kondisi
    - Difokuskan pada masing-masing kondisi dalam program
    - Bertujuan mendeteksi tidak hanya kesalahan dalam kondisi program, tetapi juga kesalahan lain
  - Pengujian aliran data
    - Menggunakan variabel-variabel pada program
  - Pengujian loop
    - Berfokus pada validitas konstruksi loop

BLACK BOX TESTING

Konsep

- Pengujian berfokus pada persayaratan fungsionalitas software
  - *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukanpengetesan pada spesifikasi fungsional program
- Black Box Testing bukanlah solusi alternatif dari White-Box Testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh White-Box Testing

Konsep

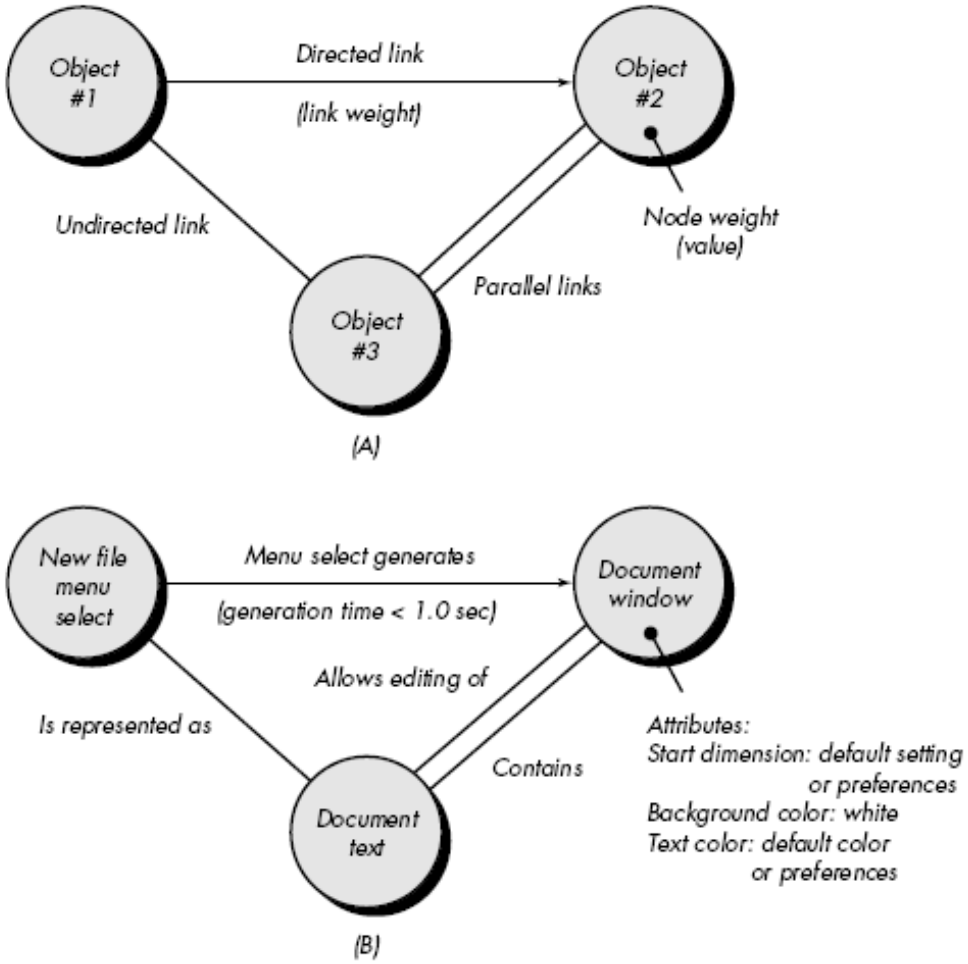
- Black-Box Testing cenderung untuk menemukan hal-hal berikut:
  - Fungsi yang tidak benar atau tidak ada
  - Kesalahan antarmuka (*interface errors*)
  - Kesalahan pada struktur data dan akses basis data
  - Kesalahan performansi (*performance errors*)
  - Kesalahan inisialisasi dan terminasi.

Konsep

- Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:
  - Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
  - Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
  - Apakah sistem sensitif pada input-input tertentu?
  - Bagaimana sekumpulan data dapat diisolasi?
  - Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
  - Efek apa yang dapat membuat kombinasi data ditangani spesifik pada
  - operasi sistem?

Black Box Testing – Graph Based

FIGURE 17.9
(A) Graph notation
(B) Simple example



Black Box Testing – Equivalence Partitioning

- Contoh: Input NPM dalam SIAMIK
  - Jika dikosongi?
  - Jika diisi dengan format yang salah?
  - Jika diisi dengan NPM yang benar?

1. If an input condition specifies a *range,* one valid and two invalid equivalence classes are defined.
2. If an input condition requires a specific *value,* one valid and two invalid equivalence classes are defined.
3. If an input condition specifies a member of a *set,* one valid and one invalid equivalence class are defined.
4. If an input condition is *Boolean,* one valid and one invalid class are defined.

Black Box Testing – Analisa Nilai Batas

1. Menguji untuk input di sekitar batas atas maupun bawah sebuah range nilai yang valid.
2. Menguji nilai maksimal dan minimal.
3. Menerapkan (1 & 2) untuk output.
4. Menguji batas struktur data yang dipakai. Misal ukuran array.

Black Box Testing – Perbandingan

- Spesifikasi kebutuhan yang sama dimungkinkan menghasilkan aplikasi/ perangkat lunak yang berbeda.
- Skenario pengujian pada aplikasi yang demikian bisa digunakan untuk skenario pengujian aplikasi serupa yang lain.

TESTING DENGAN KASUS KHUSUS

Lingkungan Pengujian

- Pengujian yang dilakukam:
  - Pengujian GUI.
  - Pengujian arsitektur client/ server.
  - Pengujian dokumentasi dan fasilitas bantuan.
  - Pengujian sistem waktu nyata