Blackbox Testing

- **Metode Graph Based Testing**
- **Metode Equivalence Partitioning**
- **Boundary Value Analysis**
- **Cause-Effect Graphing Techniques**
- **State Transition Testing**
- **Orthogonal Array Testing**
- **Functional Analysis**
- **Use Cases**

Black Box Testing

- *Black box testing*, dilakukan tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites juga disebut sebagai *behavioral testing*, *specification-based testing*, *input/output testing atau functional testing*.

- *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*.

- *Black box testing* digunakan pada tahap akhir dan berfokus pada domain informasi.

Kategori error

- Kategori *error* yang akan diketahui melalui *black box testing*:
  - Fungsi yang hilang atau tak benar
  - *Error* dari antar-muka
  - *Error* dari struktur data atau akses eksternal database
  - *Error* dari kinerja atau tingkah laku
  - *Error* dari inisialisasi dan terminasi

Tujuan Black box Testing

- Tes didisain untuk menjawab pertanyaan sebagai berikut:
  - Bagaimana validasi fungsi yang akan dites?
  - Bagaimana tingkah laku dan kinerja sistem dites?
  - Kategori masukan apa saja yang bagus digunakan untuk test cases?
  - Apakah sebagian sistem sensitif terhadap suatu nilai masukan tertentu?
  - Bagaimana batasan suatu kategori masukan ditetapkan?
  - Sistem mempunyai toleransi jenjang dan volume data apa saja?
  - Apa saja akibat dari kombinasi data tertentu yang akan terjadi pada operasi sistem?

Test case black box

- Dengan menerapkan teknik *black box*, dapat dibuat sekumpulan *test cases* yang memuaskan kriteria-kriteria sebagai berikut :

  - ***Test cases*** yang mengurangi jumlah *test cases* (lebih dari satu) yang didisain untuk mencapai testing yang masuk akal.

  - ***Test cases*** yang dapat memberikan informasi tentang kehadiran kelas-kelas dari *error*.

Dekomposisi kebutuhan untuk dites secara sistematis (1)

- Karakteristik suatu testcase:
  - Dekomposisi dari tugas-tugas testing suatu sistem ke aktivitas-aktivitas yang lebih kecil
  - Dapat dimanajemeni,
  - Tercapai *test case* individual
  - *test case* yang ada telah cukup mencakup semua aspek dari sistem
  - Pendisainan *test case* dilakukan secara manual

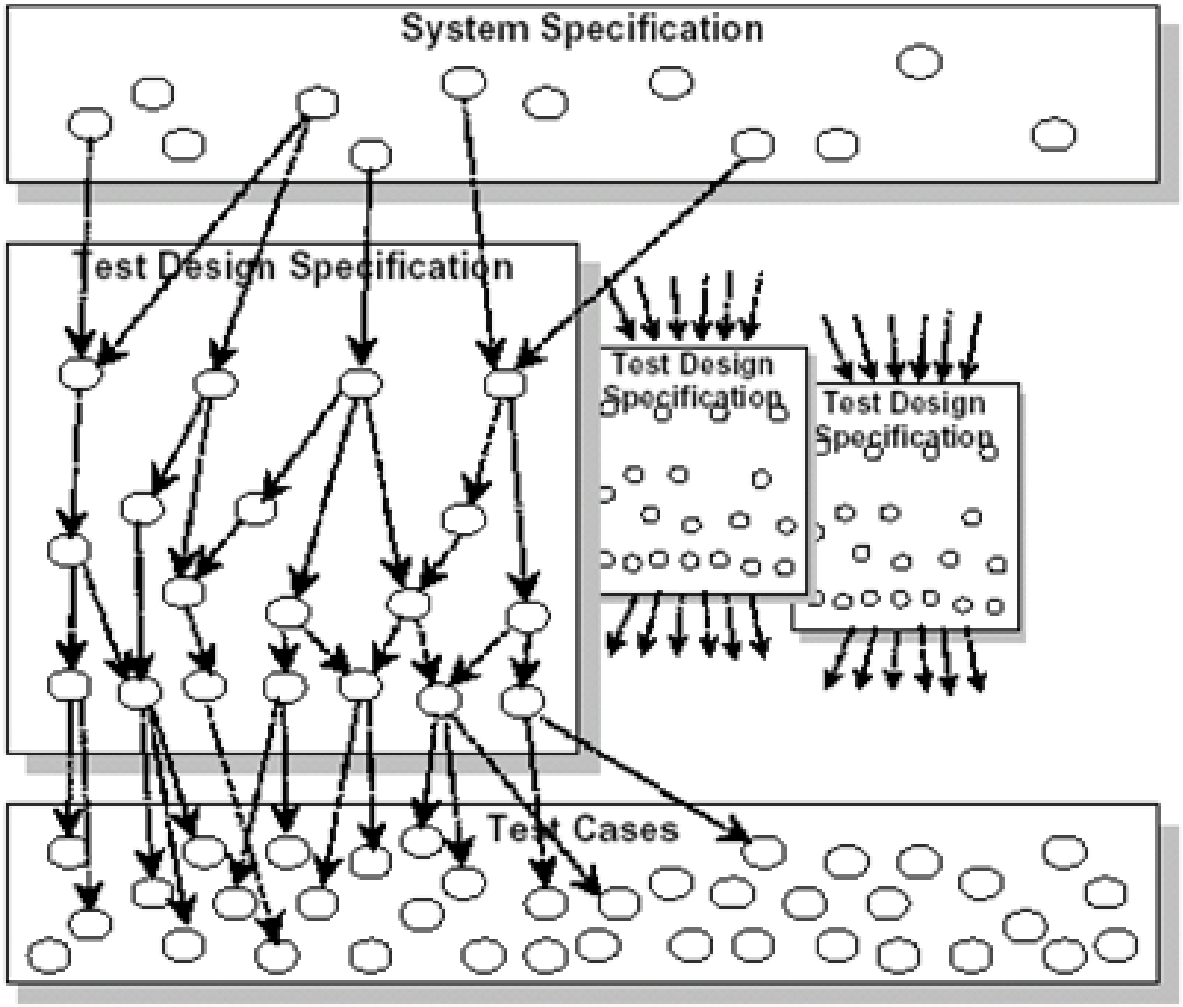
Dekomposisi kebutuhan untuk dites secara sistematis (2)

- **Spesifikasi sebagai tuntunan testing**
  - Spesifikasi atau model sistem adalah titik awal dalam memulai disain tes.
  - Spesifikasi atau model sistem dapat berupa spesifikasi fungsional, spesifikasi kinerja atau keamanan, spesifikasi skenario pengguna, atau spesifikasi berdasarkan pada resiko sistem.
  - Spesifikasi menggambarkan kriteria yang digunakan untuk menentukan operasi yang benar atau dapat diterima, sebagai acuan pelaksanaan tes.

Dekomposisi kebutuhan untuk dites secara sistematis (3)

- **Dekomposisi obyektifitas tes**
  - Disain tes berfokus pas tda spesifikasi komponen yang dites.
  - Obyektifitas tes tingkat atas disusun berdasarkan pada spesifikasi komponen.
  - Tiap obyektifitas tes ini untuk kemudian didekomposisikan ke dalam obyektifitas tes lainnya atau *test cases* menggunakan teknik disain tes.

Dekomposisi obyektifitas tes



Dekomposisi tugas testing adalah memecah-mecah sistem ke obyektifitas-obyektifitas tes yang lebih kecil cakupannya hingga sekumpulan *test cases* teridentifikasi.

Tipe testing Blackbox

Terdapat banyak jenis teknik disain tes yang dapat dipilih berdasarkan pada tipe testing yang akan digunakan yaitu:

- *Equivalence Class Partitioning*
- *Boundary Value Analysis*
- *State Transitions Testing*
- *Cause-Effect Graphing*

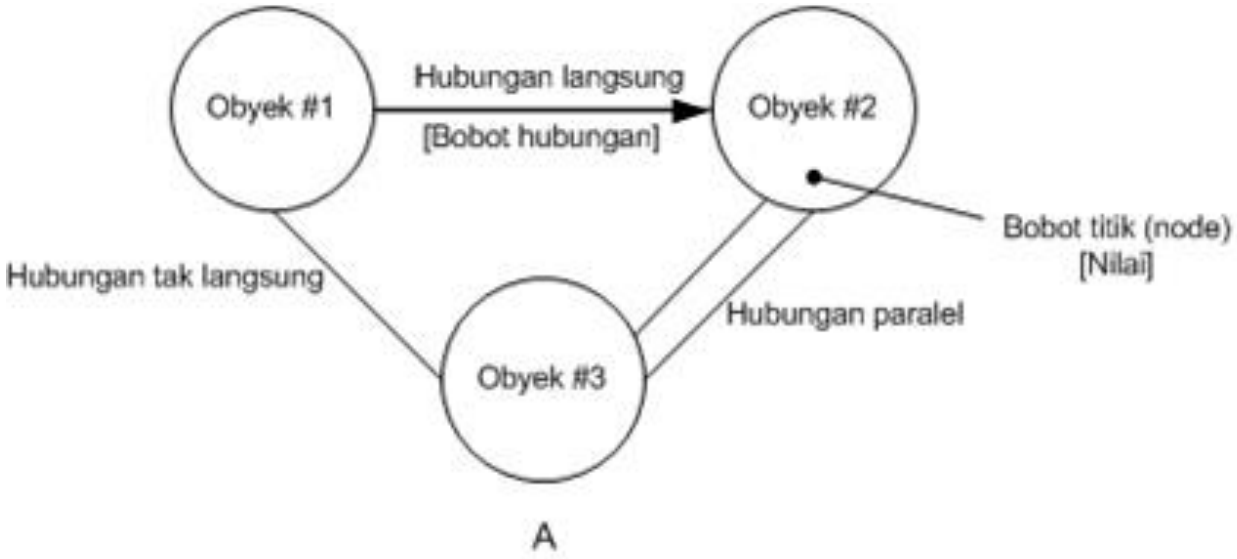
I. Metode Graph Based Testing

- **Langkah pertama:**
  Memahami obyek yang dimodelkan dalam *software* dan hubungan koneksi antar obyek.

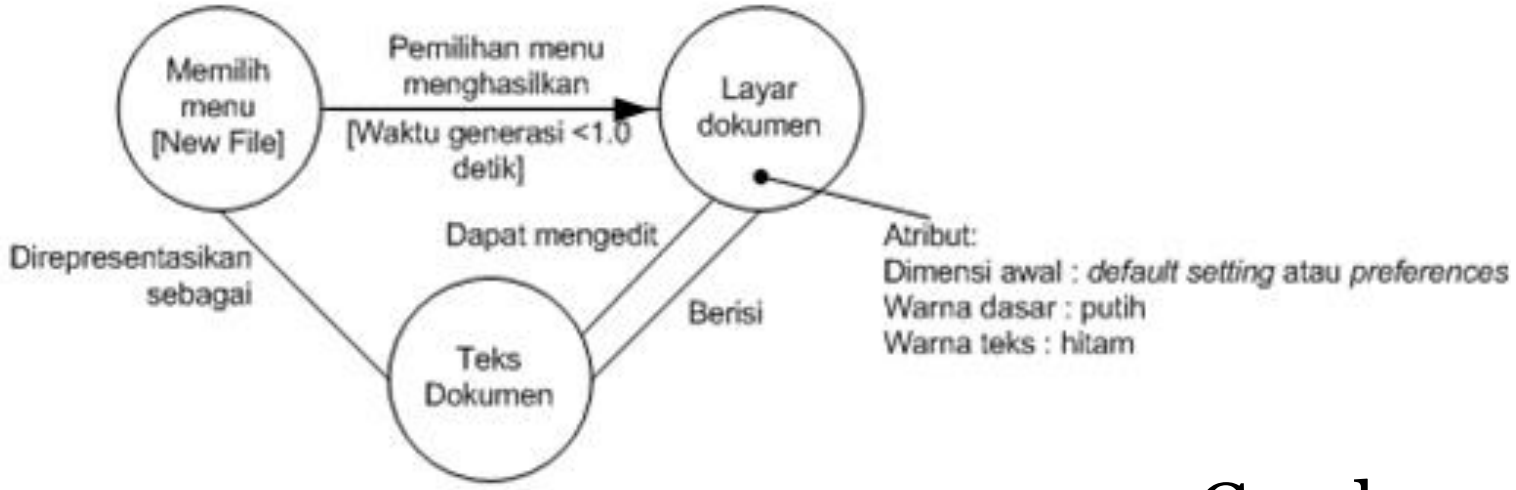
- **Langkah kedua:**
  Mendefinisikan serangkaian tes yang merupakan verifikasi bahwa semua obyek telah mempunyai hubungan dengan yang lainnya sesuai yang diharapkan.

- Pada langkah kedua ini dapat dicapai dengan membuat grafik, dimana berisi:

  - Kumpulan *node* yang mewakili obyek,
  - Penghubung / *link* yang mewakili hubungan antar obyek
  - Bobot *node* yang menjelaskan properti dari suatu obyek
  - Bobot penghubung yang menjelaskan beberapa karakteristik dari penghubung / *link*.


Obyek #1
Hubungan langsung
[Bobot hubungan]
Obyek #2
Bobot titik (node)
[Nilai]
Hubungan tak langsung
Hubungan paralel
Obyek #3

A


Memilih menu [New File]
Pemilihan menu menghasilkan
[Waktu generasi <1.0 detik]
Layar dokumen
Direpresentasikan sebagai
Dapat mengedit
Berisi
Teks Dokumen
Atribut:
Dimensi awal : default setting atau preferences
Warna dasar : putih
Warna teks : hitam

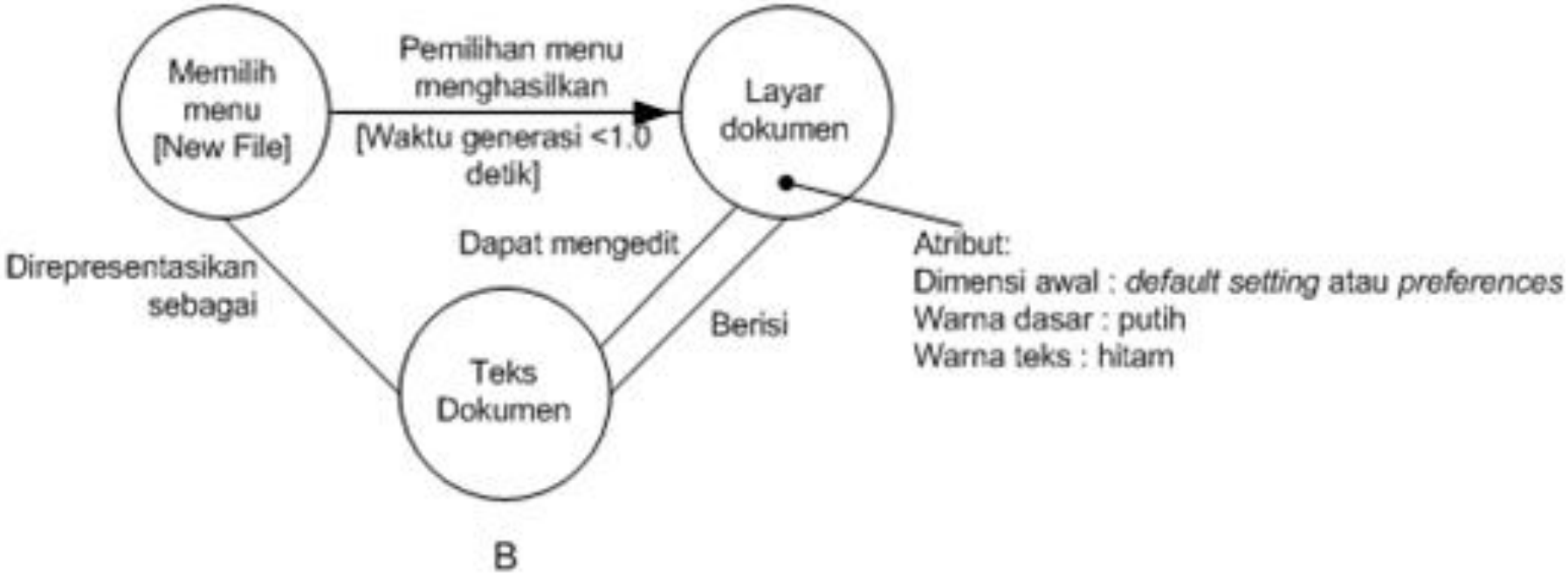
B

Gambar. 1

Keterangan

- ***Nodes*** direpresentasikan sebagai lingkaran yang dihubungkan dengan garis penghubung.
- ***Hubungan langsung (digambarkan dalam bentuk anak panah)***, mengindikasikan suatu hubungan yang bergerak hanya dalam satu arah.
- ***Hubungan dua arah***, juga disebut sebagai hubungan simetris, menggambarkan hubungan yang dapat bergerak dalam dua arah.
- ***Hubungan paralel***, digunakan bila sejumlah hubungan ditetapkan antara dua *nodes*.

Contoh Ilustrasi (Gambar. 1)



B

Contoh Ilustrasi (Gambar. 1)

- Dimana:
  - Obyek #1 = Memilih menu [New File].
  - Obyek #2 = Layar dokumen.
  - Obyek #3 = Teks dokumen.

- Pemilihan menu [New File] akan menghasilkan (*generate*) layar dokumen.
- Bobot hubungan mengindikasikan bahwa layar harus telah dibuat dalam waktu kurang dari 1 detik.
- Suatu hubungan tak langsung antara pemilihan menu [New File] dengan teks dokumen.
- Hubungan paralel mengindikasikan hubungan layar dokumen dan teks dokumen.

Metode tingkah laku testing yang dapat menggunakan grafik:

A. **Pemodelan Alur Transaksi**
B. **Pemodelan *Finite State***
C. **Pemodelan Alur Data**
D. **Pemodelan Waktu / *Timing***

A. Pemodelan Alur Transaksi

- ***Node*** mewakili langkah-langkah transaksi
  - Misal: langkah-langkah penggunaan jasa reservasi tiket pesawat secara *on-line*

- ***Penghubung*** mewakili logika koneksi antar langkah.
  - Misal: masukan informasi penerbangan diikuti dengan pemrosesan validasi / keberadaan.

B. Pemodelan *Finite State*

- ***node*** mewakili status *software* yang dapat diobservasi.
  - Misal: tiap layar yang muncul sebagai masukan order ketika kasir menerimaa order

- ***penghubung*** mewakili transisi yang terjadi antar status.
  - Misal: informasi order diverifikasi dengan menampilkan keberadaan inventori dan diikuti dengan masukan informasi penagihan pelanggan

C. Pemodelan Alur Data

- ***node*** mewakili obyek data.
  - Misal: data Pajak dan Gaji Bersih

- ***penghubung*** mewakili transformasi untuk me-translasikan antar obyek data.
  - Misal: Pajak = 0.15 x Gaji Bersih).

D. Pemodelan Waktu / *Timing*

- ***Node*** mewakili obyek program.
- ***Penghubung*** mewakili sekuensial koneksi antar obyek tersebut
- ***Bobot penghubung*** digunakan untuk spesifikasi waktu eksekusi yang dibutuhkan.

Tahapan testing berbasis grafik

1. Mendefinisikan semua *nodes*
   - Obyek dan atribut didefinisikan terlebih dahulu.
   - *Data model* dapat digunakan sebagai titik awal untuk memulai
   - kebanyakan *nodes* merupakan obyek dari program (yang tidak secara eksplisit direpresentasikan dalam *data model*).
2. Menetapkan Hubungan
   - Hubungan harus diberi nama
3. Menetapkan Bobot Hubungan