



# IF2250 Dasar Rekayasa Perangkat Lunak

## Pengujian Perangkat Lunak (2): Integration Testing

Oleh: Bayu Hendradjaya

Pengajar:

Bayu Hendradjaya/Christine Suryadi



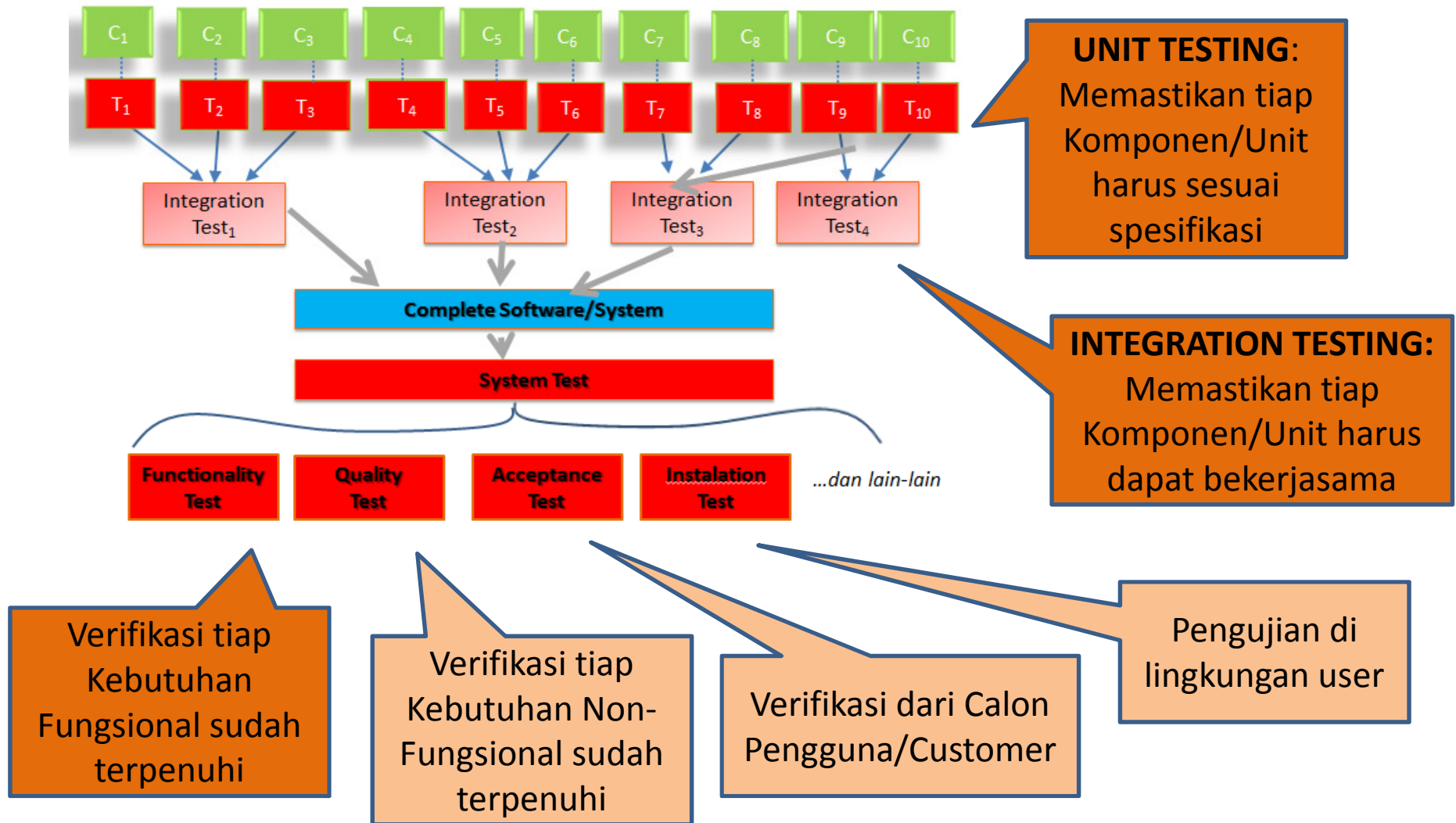
*Slide kuliah bisa didownload dari  
[kuliah.itb.ac.id](http://kuliah.itb.ac.id) (subyek IF2250)*



# Review



# Peran tiap pengujian

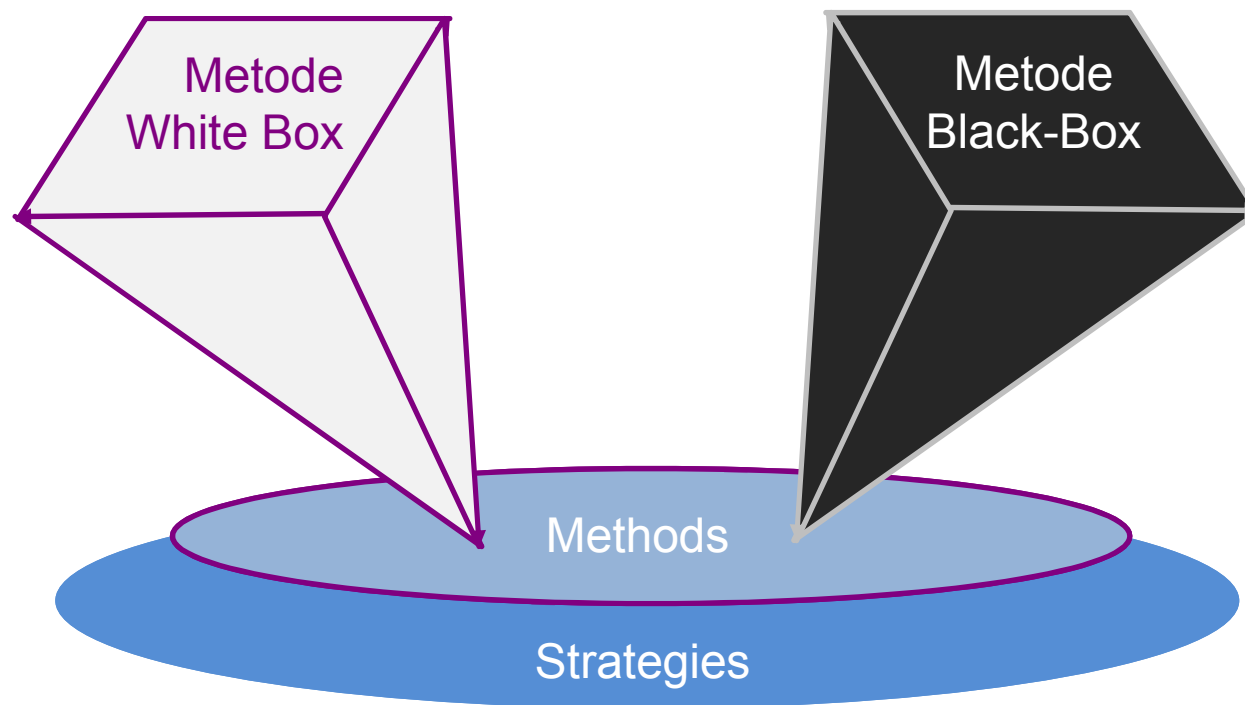


# Software Testing



Penguji menggunakan **Source Code** untuk mengembangkan Kasus Uji (Test Case)

Penguji tidak menggunakan Source Code, tetapi memanfaatkan **spesifikasi** untuk membuat Kasus Uji (Test Case)





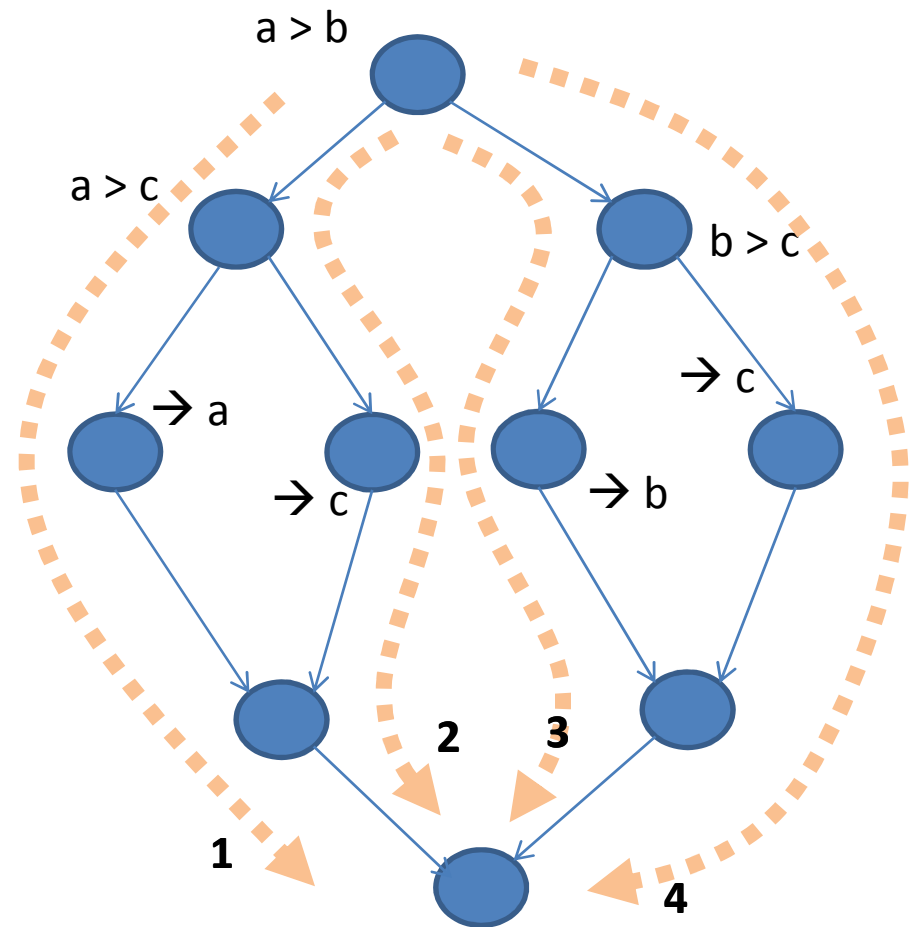
# Contoh Strategi Pengujian WHITE BOX :

## “Control Flow Testing”



# CFG<sub>Graph</sub> dapat disederhanakan menjadi kumpulan node

```
if a > b then  
  if a > c then  
    → a  
  else  
    → c  
else  
  if b > c then  
    → b  
  else  
    → c
```



# Apa Guna Control Flow Testing

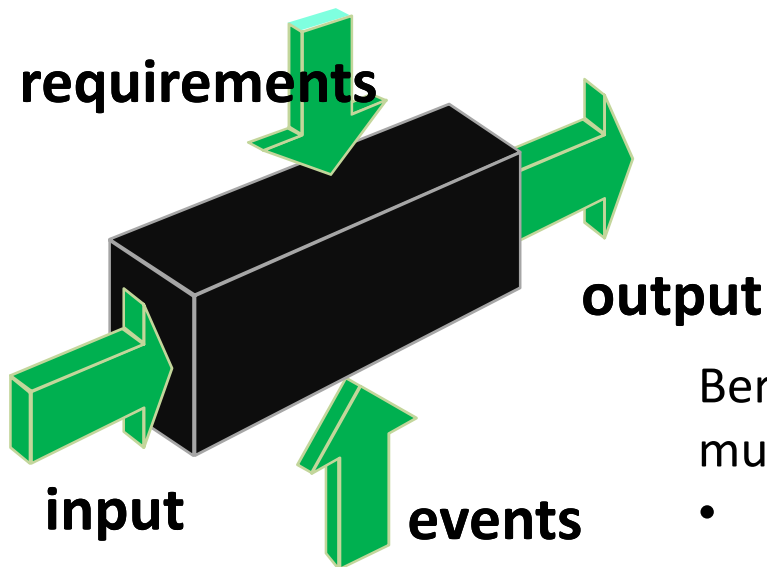


- Caranya:
  - Buat Control Flow Graph
  - Tetapkan jalur yang harus di uji
  - Cari test case yang akan melalui jalur tadi minimal satu kali.
  - Tetapkan nilai yang diharapkan (*expected outcome/expected result*)





# Black Box Testing



*Pengujian Black-Box dilakukan karena isi source code tidak bisa kita lihat, hanya spesifikasinya saja!*

*Disebut juga **Functional Testing** karena kita hanya melihat fungsionalitas dari unit yang diuji*

Berdasarkan spesifikasi requirement (yang mungkin melibatkan event):

- *tentukan sejumlah input,*
- *lalu tentukan output yang diharapkan.*





# Equivalence Class Partitioning (ECP) & Boundary Value Analysis (BVA)



# ECP dan BVA



- Jadi secara umum teknik yang dilakukan:
  1. Partisi nilai input
  2. Pilih satu data dari setiap partisi
  3. Pilih data di titik batas





## Contoh lain

- Test program yang menampilkan bilangan terbesar antara dua input. Kedua input selalu dalam range 0-10000. Jika nilainya sama, maka akan ditampilkan nilainya yang sama.
  - Maka kita bisa mengambil nilai sembarang antara 0-10000 untuk keduanya:
    - Misalnya 500 dan 1000
  - Kemudian diambil nilai batas, 0 dan 10000

# Kasus Uji



Nomor Test	Angka 1	Angka 2	Hasil yang diharapkan
1	0	0	0
2	0	1000	1000
3	0	10000	10000
4	500	0	500
5	500	1000	1000
6	500	10000	10000
7	10000	0	10000
8	10000	1000	10000
9	10000	10000	10000





# Integration Testing



# Pengujian Integrasi (Integration Testing)

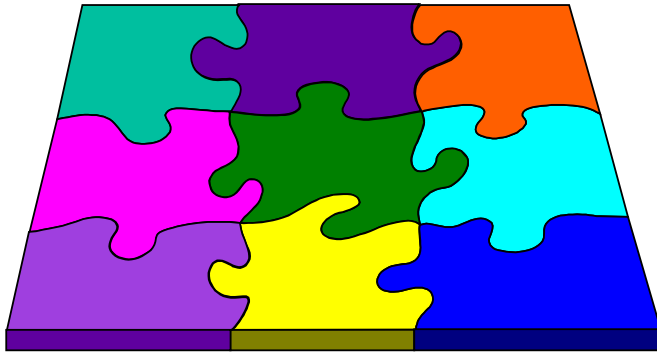


- Pengujian unit hanya fokus pada komponen/unit individual
  - Sesudah semua fault/bug/defect dihilangkan dari setiap unit, maka unit/komponen ini siap diintegrasikan atau digabungkan
- Proses integrasi unit-unit tadi akan memungkinkan terjadinya error yang tidak terdeteksi sebelumnya.
  - Pengujian integrasi dapat menemukan kesalahan atau bahkan dapat mendeteksi fault/bug/defect yang belum terdeteksi saat pengujian unit
  - Pengujian integrasi fokusnya lebih pada sekumpulan komponen/unit
- Idealnya, proses integrasi atau penggabungan, dilakukan satu persatu, setiap kali penggabungan satu unit maka akan dilakukan pengujian.
  - Jika kesalahan tidak ditemukan, maka unit lain akan digabungkan, hingga seluruh unit terintegrasi
  - Strategi penggabungan/integrasi akan menentukan lama tidaknya pengujian integrasi dilakukan
    - Strategi yang salah akan menambah biaya dan waktu



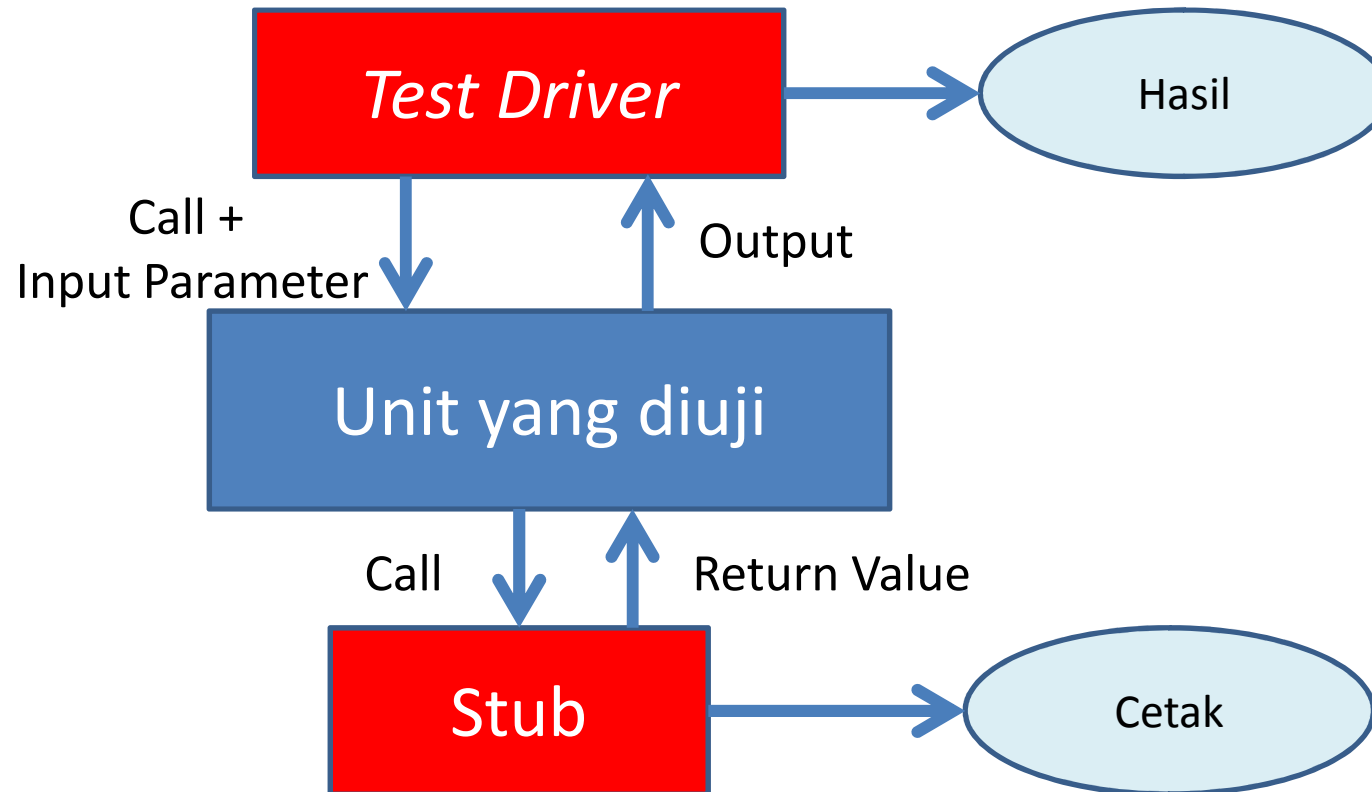


# Strategi Pengujian Integrasi



- Pendekatan incremental
  - Top Down testing
  - Bottom Up testing
  - Sandwich Testing
- Pendekatan 'BIG BANG'

# Review: Lingkungan Pengujian







## Unit Under Test

```
int HitungGajiPegawai(int NIP)
{
    int GaPok = ReadGajiPokok(NIP);
    int GaTun = ReadGajiTun( NIP);

    return GaPok + GaTun;
}
```

## STUB

```
int ReadGajiPokok(int NIP)
{
    // execute sql statement
    // select gaji from Pegawai
    return 2500;
}
```

```
int ReadGajiTun(int NIP)
{
```

```
    // execute sql statement
    // select tunj from Pegawai
    return 7500;
}
```

## Test Driver

```
int main()
{
    assert(HitungGajiPegawai(111), 10000);
    assertNotTrue(HitungGajiPegawai(200), 5000);
}
```

# Strategi Top Down Testing



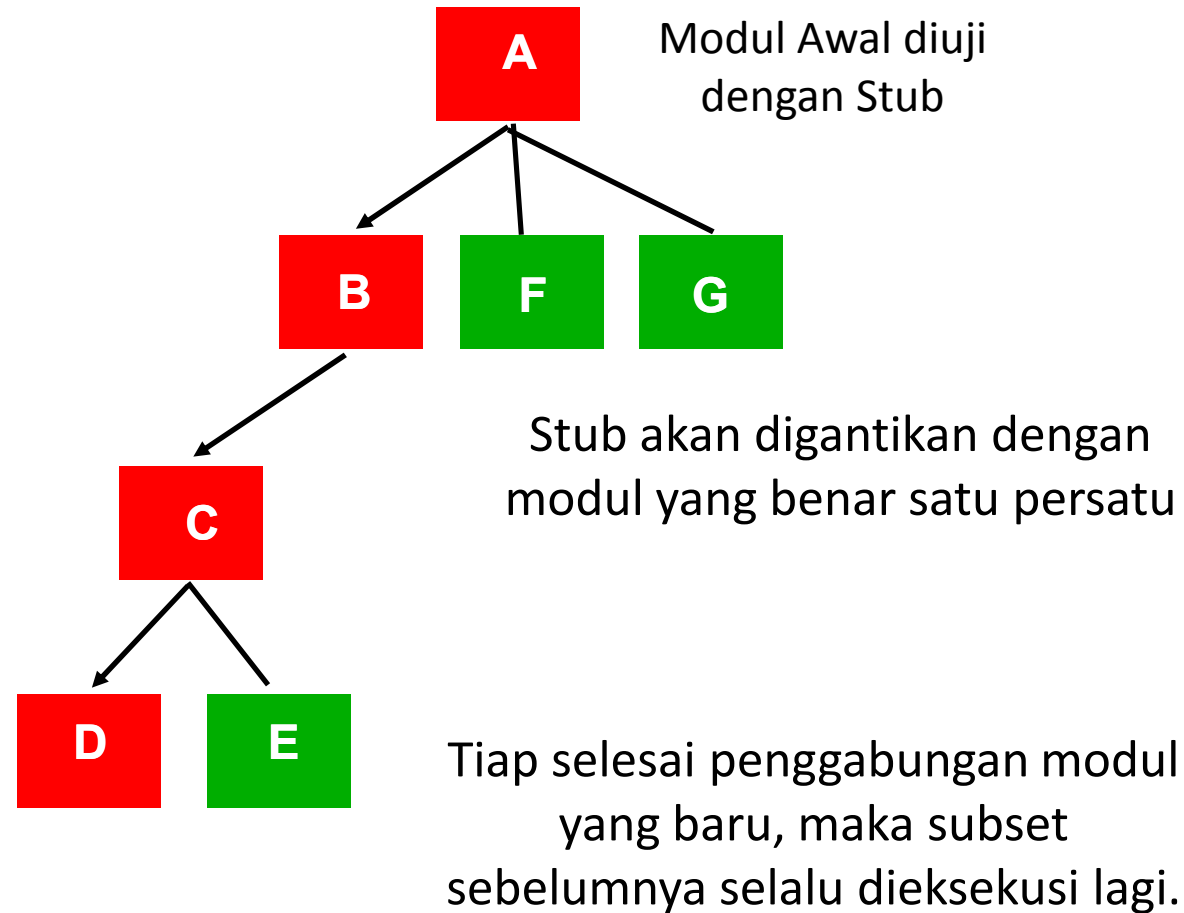
- Pengujian dimulai dari komponen level atas, biasanya bagian User Interface (UI), lalu diikuti komponen-komponen di bawahnya, hingga semua sudah diuji.
- Butuh Stub untuk komponen di bawahnya, test driver tidak diperlukan

**Keuntungan:** Biasanya dimulai dari UI, (atau menu) atau dari program utama sehingga mudah (tidak butuh test driver)


**Kerugian:** kadang banyak stub dibutuhkan, padahal pengembangan stub butuh waktu dan stub sering ada error




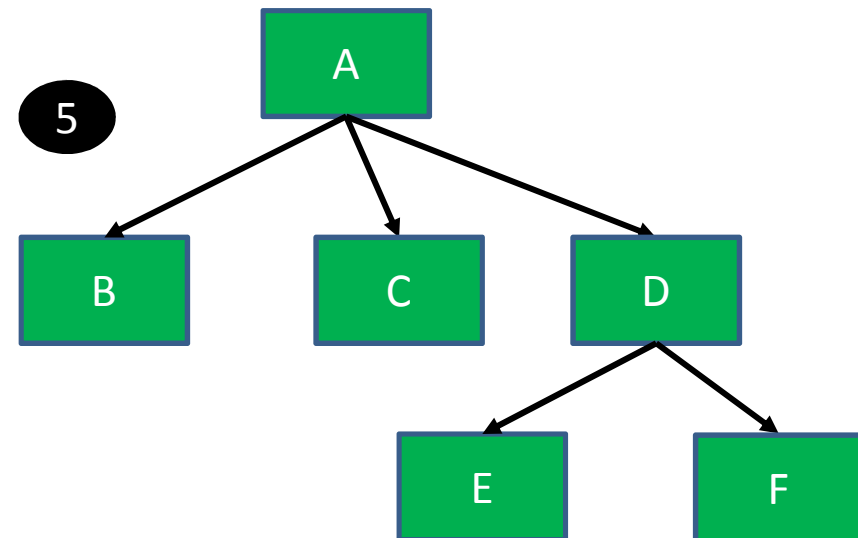
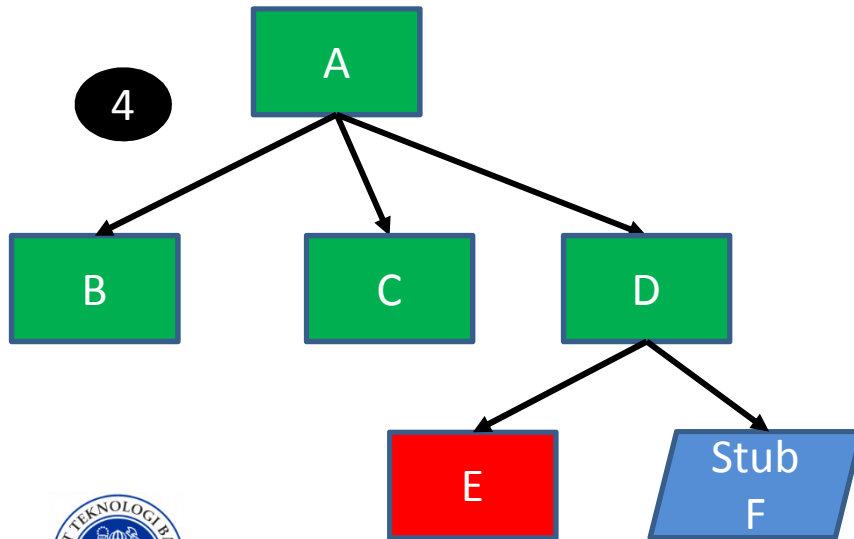
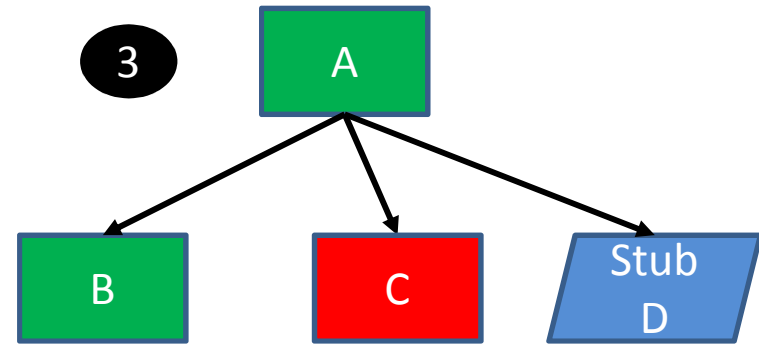
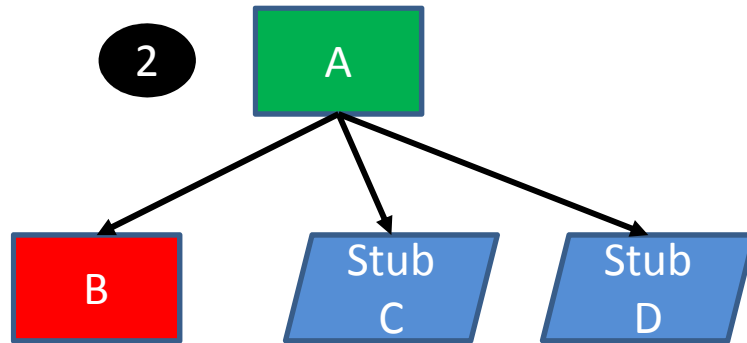
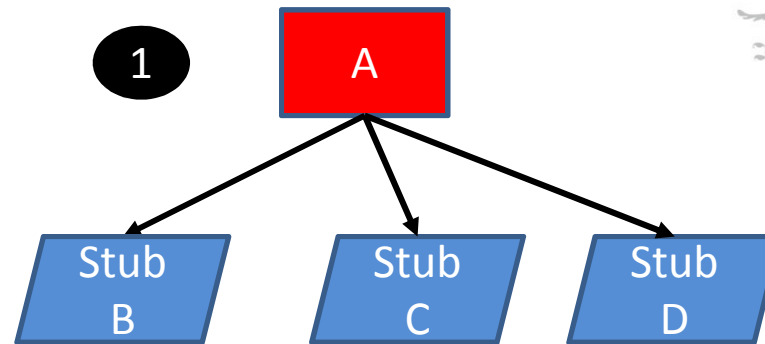
# Strategi Top Down Testing (2)



# Top Down Testing

 Modul yang baru digabung, perlu diuji

 Modul yang sudah lolos uji



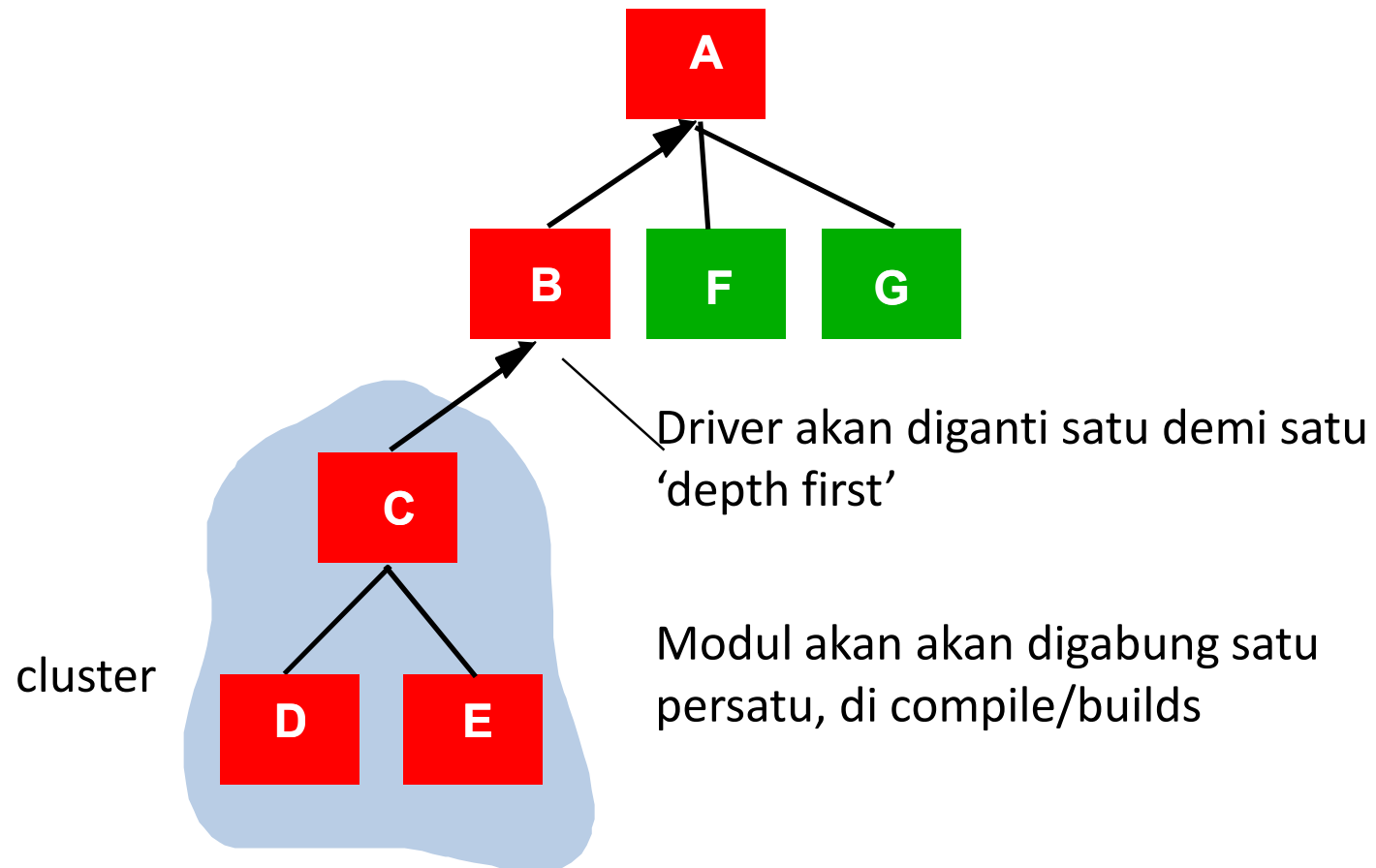
# Strategi Bottom-Up Testing



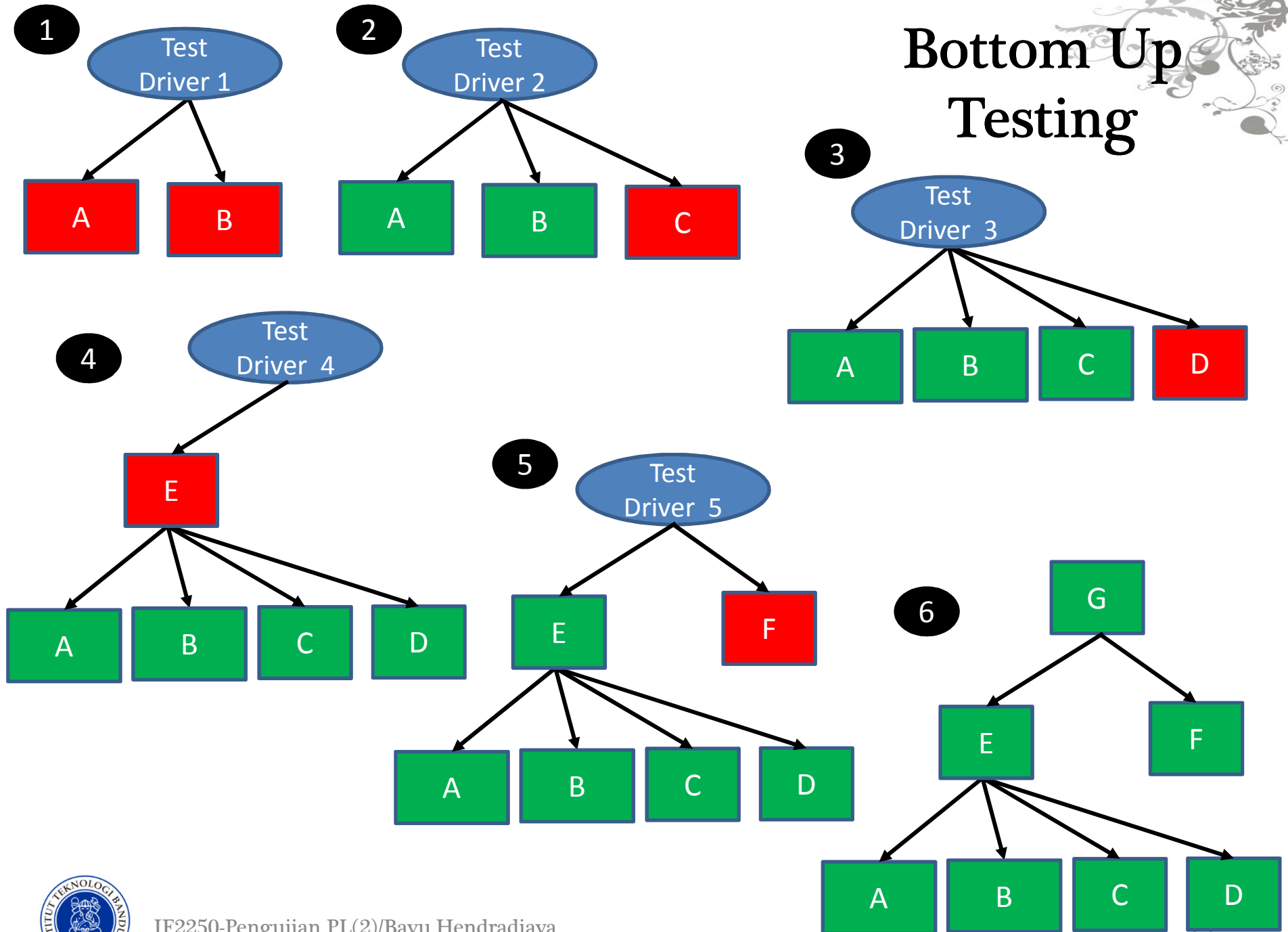
- Setelah tiap komponen/unit di lapisan bawah di lakukan unit-test, maka komponen/unit ini diintegrasikan dengan komponen di lapisan atasnya.
  - Diulang hingga semua level di atasnya terintegrasi
  - Butuh test driver untuk men-simulasikan komponen/unit di level atasnya
    - Test Stub tidak diperlukan



# Strategi Bottom-Up Testing (2)



# Bottom Up Testing



# Strategi Pengujian Sandwich (Sandwich Testing)

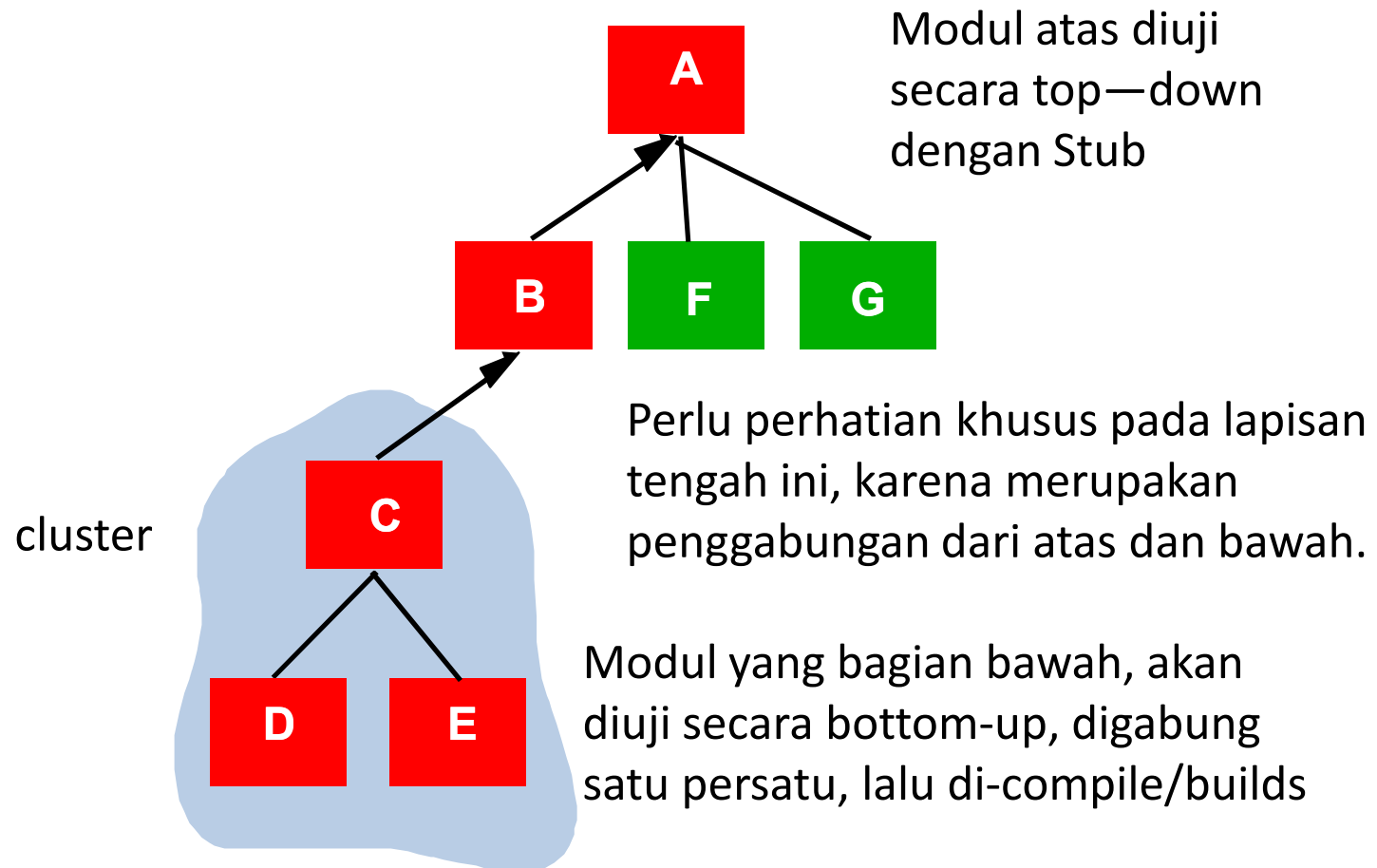


- Penggabungan strategy Top-down dan Bottom-Up
  - Sistem seolah dibagi menjadi tiga lapisan
    - Lapisan atas
    - Lapisan bawah
    - Lapisan target (yang terletak antara lapisan atas dan bawah)
  - Dengan fokus pada lapisan target, maka top-down/bottom-up testing dilakukan secara bersamaan (paralel)





# Pengujian 'Sandwich' (Sandwich Testing)



# Pengujian Regresi (Regression Testing)



- Pengujian Regresi melakukan re-eksekusi terhadap beberapa subset test yang sudah dilakukan sebelumnya untuk meyakinkan bahwa perubahan yang baru tidak berakibat pada hasil pengujian sebelumnya
- Jika software diperbaiki, beberapa aspek dalam konfigurasi juga perlu diperbaiki (dokumentasi, program, ataupun data)
- Pengujian regresi akan meyakinkan bahwa perubahan tidak memberikan perilaku yang tidak diinginkan atau tidak mempengaruhi modul-modul sebelumnya.
- Pengujian regresi dapat dilakukan secara manual dengan mengeksekusi ulang semua kasus uji tetapi menggunakan tools akan memungkinkan otomatisasi sehingga akan lebih cepat dan mudah
  - Misalnya penggunaan JUnit (untuk java), NUnit (.NET) atau PUnit (Php).
- Pengujian Regresi juga perlu dilakukan untuk Unit Testing



# Big Bang Testing



- Pada Big Bang testing, pengujian dengan langsung menggabung semua unit/komponen tadi sebagai sistem yang utuh
  - **Keuntungan:** tidak perlu stub atau driver
  - **Kerugian:**
    - Bila terjadi suatu error mungkin sulit dicari bug-nya
      - Sulit diketahui apakah faultnya terletak di interface atau terletak di bagian dalam unit/komponenennya.
- Big Bang Testing bukanlah cara yang ideal, nampaknya cepat, tetapi untuk software skala menengah hingga besar, cara ini tidak di rekomendasikan.





# Pengujian Kebutuhan Fungsional *atau* Pengujian Fungsional *(Functional Testing)*



# Functional Testing



- Functional Testing adalah bagian dari System Testing
  - Contoh system testing lain: performance testing, security testing, recovery testing, acceptance testing, pilot testing dan installation testing
- Fokus dari Pengujian fungsional adalah menjamin semua kebutuhan fungsional sudah diuji
  - Hasil implementasi program harus sudah mencakup semua yang dituliskan dalam spesifikasi kebutuhan perangkat lunak
- Pengujian ini sifatnya adalah pengujian **Black-Box**
  - Memanfaatkan model use case
    - Dipilih suatu instansiasi dari use case yang kemungkinan bisa menyebabkan error
      - Bisa memanfaatkan Equivalence Class Partitioning/Boundary Testing
      - Test case dibuat berdasarkan skenario normal dan juga skenario alternatif

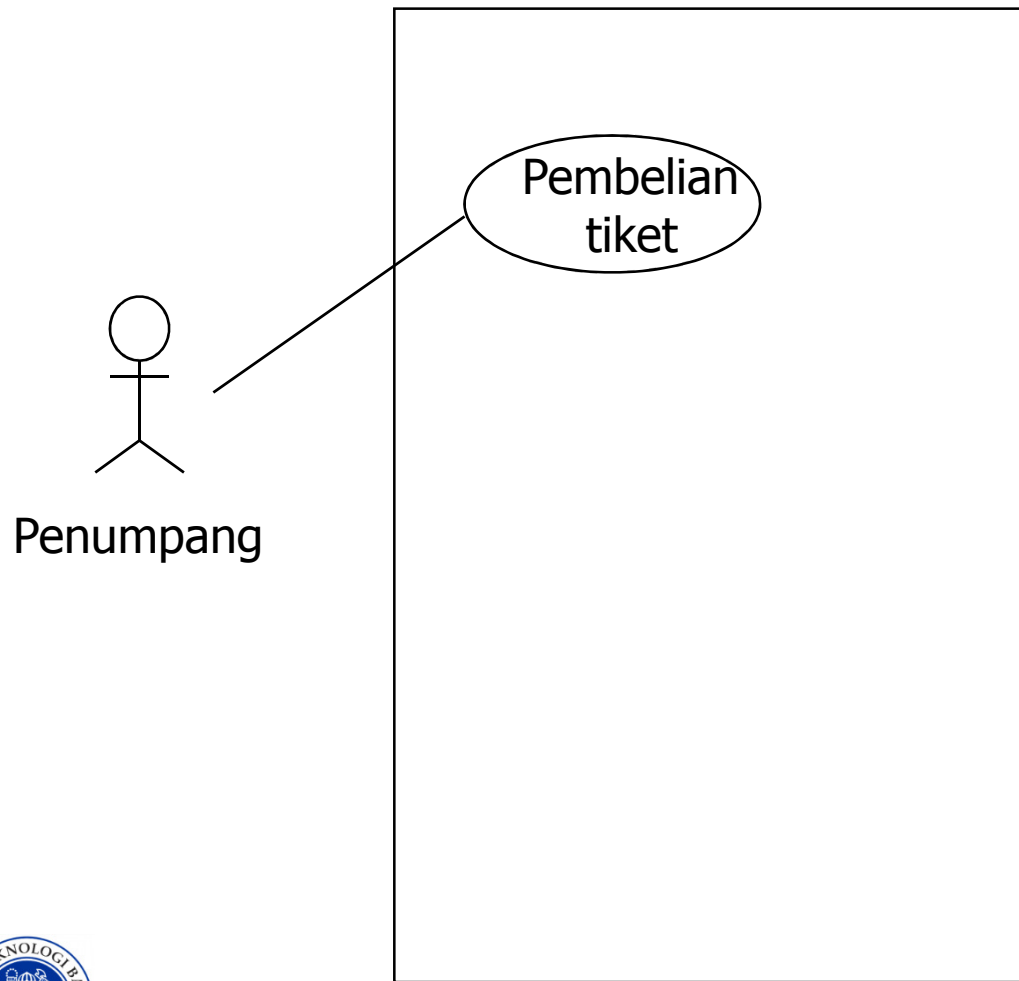


# Studi Kasus: Mesin Tiket Kereta Api





# Contoh use case pembelian karcis kereta lewat mesin tiket



Diketahui ada 5 stasiun tujuan, dengan biaya masing-masing

- stasiun 1: 1000,
- stasiun 2: 1500 ,
- stasiun 3: 2500,
- stasiun 4: 3000,
- stasiun 5: 4000

# Use case: Pembelian Tiket



**Kondisi Awal:** Penumpang berdiri di depan mesin tiket dengan uang yang cukup

## **Skenario Normal:**

- Penumpang memilih tujuan stasiun, dengan menekan tombol tujuan, bila ada lebih dari satu yang ditekan, maka pilihan terakhir yang diambil
- Mesin akan menampilkan biaya yang harus dibayar
- Penumpang memasukkan koin ke slot mesin
  - Alternatif skenario
    - Jika penumpang memasukkan stasiun baru sebelum uang yang dimasukkan cukup, maka mesin akan mengembalikan seluruh koin
    - Jika penumpang memasukkan jumlah koin lebih, maka pengembalian akan diberikan oleh mesin
- Mesin akan mencetak tiket
- Penumpang akan mengambil tiket dan kembalian jika ada

**Kondisi akhir:** Tiket sudah ditangan penumpang







# Ide pengujian

- Dari use-case tersebut, maka ada beberapa fitur dari mesin tiket tersebut yang harus diuji
  - Jika penumpang menekan tombol dan memasukkan jumlah koin yang tepat, maka ia akan mendapatkan tiket saja.
  - Jika penumpang menekan tombol untuk beberapa stasiun, maka mesin harus menampilkan harga tiket stasiun terakhir
  - Jika penumpang menekan tombol stasiun lain, selagi memasukkan uang ke mesin, maka uang otomatis dikembalikan dulu
  - Jika penumpang memasukkan uang yang melebihi yang harus dibayar, mesin harus melakukan pengembalian



# Test Case 1



Nama Test case	TC1: Uang Pas
Kondisi awal	Penumpang di depan mesin, dan memiliki koin 5 koin 1000, 2 koin 500
Skenario	<ol style="list-style-type: none"><li>1. Penumpang menekan tombol stasiun 3 dan mesin akan menampilkan 2500</li><li>2. Penumpang memasukkan dua koin 1000 dan satu koin 500</li></ol>
Kondisi akhir	Penumpang mendapatkan tiket untuk stasiun 3 dan tidak ada koin kembalian



# Test Case 2



Nama Test case	TC2: Penekanan tombol berbeda
Kondisi awal	Penumpang di depan mesin, dan memiliki koin 5 koin 1000, 2 koin 500
Skenario	<ol style="list-style-type: none"><li>1. Penumpang menekan tombol stasiun 5, 3, 1, 2, pada saat yang bersamaan setiap kali mesin akan menampilkan 4000, 2500, 1000, lalu 1500</li><li>2. Penumpang memasukkan dua koin 1000</li><li>3. Mesin harus mengembalikan koin sejumlah 500</li></ol>
Kondisi akhir	Penumpang mendapat tiket ke stasiun 2 dan mendapatkan pengembalian koin 500



# Test Case 3



Nama Test case	TC3: pemilihan stasiun lain saat koin sudah dimasukkan
Kondisi awal	Penumpang di depan mesin, dan memiliki koin 5 koin 1000, 2 koin 500
Skenario	<ol style="list-style-type: none"><li>1. Penumpang menekan tombol stasiun 5 dan mesin akan menampilkan 4000</li><li>2. Penumpang memasukkan dua koin 1000</li><li>3. Lalu penumpang menekan tombol 3</li><li>4. Mesin harus langsung mengembalikan koin sejumlah 2000</li></ol>
Kondisi akhir	Penumpang mendapat Koin kembali sejumlah 2000



# Test Case 4



Nama Test case	TC4: Pengembalian Koin
Kondisi awal	Penumpang di depan mesin, dan memiliki koin 5 koin 1000, 2 koin 500
Skenario	<ol style="list-style-type: none"><li>1. Penumpang menekan tombol stasiun 3 dan mesin akan menampilkan 2500</li><li>2. Penumpang memasukkan tiga koin 1000</li><li>3. Mesin harus mengembalikan koin sejumlah 500</li></ol>
Kondisi akhir	Penumpang mendapatkan tiket dan koin kembali sejumlah 500





# Dokumen Pengujian

## Penjelasan tambahan untuk tugas



# Cover

**GL03**

**PERENCANAAN, DESKRIPSI, DAN HASIL  
UJI PERANGKAT LUNAK**

**<Nama Perangkat Lunak>**

untuk:


**<Nama User>**

Dipersiapkan oleh:

**<Nomor Grup & Anggota>**

Departemen Teknik Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Departemen Teknik Informatika ITB	Nomor Dokumen		Halaman
		<b>GL03-GXX</b> <xx:no grp>		<#>/<jml #
		Revisi	<revisi>	Tgl: <isi tanggal>



# Halaman 2

## DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								

Departemen Teknik Informatika ITB PDHUPL-Gxx Halaman 2 dari 12 halaman

Template Dokumen ini dan informasi yang dimilikinya adalah milik Departemen Teknik Informatika-ITB dan bersifat rahasia.  
Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Departemen Teknik Informatika ITB.







### Daftar Halaman Perubahan



Halaman	Revisi	Halaman	Revisi





## Daftar Isi

1	Pendahuluan.....	6
1.1	Tujuan Pembuatan Dokumen .....	6
1.2	Deskripsi Umum Sistem .....	6
1.3	Deskripsi Dokumen (Ikhtisar) .....	6
1.4	Definisi dan Singkatan.....	6
1.5	Dokumen Referensi.....	6
2	Lingkungan Pengujian Perangkat Lunak .....	6
2.1	Perangkat Lunak Pengujian.....	7
2.2	Perangkat Keras Pengujian .....	7
2.3	Material Pengujian.....	7
2.4	Sumber Daya Manusia.....	7
2.5	Prosedur Umum Pengujian .....	7
2.5.1	Pengenalan dan Latihan.....	7
2.5.2	Persiapan Awal.....	7
2.5.2.1	Persiapan Prosedural .....	8
2.5.2.2	Persiapan Perangkat Keras .....	8
2.5.2.3	Persiapan Perangkat Lunak.....	8
2.5.3	Pelaksanaan.....	9
2.5.4	Pelaporan Hasil.....	9
3	Identifikasi dan Rencana Pengujian .....	9
4	Deskripsi dan Hasil Uji.....	9
4.1	<Identifikasi kelas pengujian XXXXX>.....	10
4.1.1	<Identifikasi butir pengujian YYYYY> .....	10
4.1.2	<Identifikasi butir pengujian YYYYY> .....	10





**Daftar Gambar**

**Daftar Tabel**

**Daftar Lampiran**

Departemen Teknik Informatika ITB	PDHUPL-Gxx	Halaman 5 dari 12 halaman
Template Dokumen ini dan informasi yang dimilikinya adalah milik Departemen Teknik Informatika-ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Departemen Teknik Informatika ITB.		





## **1 Pendahuluan**

### **1.1 Tujuan Pembuatan Dokumen**

Bagian ini berisi penjelasan ringkas untuk siapa dokumen ini dibuat dan digunakan untuk apa oleh penerima dokumen ini.

Pada bagian ini juga akan dinyatakan bahwa PDHUPL ini akan digunakan untuk menguji seluruh sistem.

### **1.2 Deskripsi Umum Sistem**

Bagian ini akan berisi penjelasan ringkas sistem yang akan diuji. Tuliskan dalam satu paragraf gambaran umum sistem yang akan dibuat. Gambaran umum berisi deskripsi singkat perangkat lunak serta identifikasi perangkat lunak (nama, nomor identifikasi bila ada, judul, singkatan, nomor versi, dan nomor *release*). [Sama dengan bagian Deskripsi Umum Sistem di SKPL dan DPPL]

### **1.3 Deskripsi Dokumen (Ikhtisar)**

Bagian ini merangkum isi dan sistematika penulisan dokumen sesuai yang dengan *template* yang diberikan.





#### **1.4 Definisi dan Singkatan**

Berisi penjelasan terhadap semua definisi, akronim, dan singkatan yang digunakan atau disebutkan dalam PDHUPL yang dibuat.

#### **1.5 Aturan Penamaan dan Penomoran**

*Tuliskan aturan penomoran dan penamaan yang dipakai dalam dokumen ini jika ada (misalnya aturan penomroan Fungsi/CSU, penomoran modul, penamaan file, dsb)*

#### **1.6 Dokumen Referensi**

Informasi ini dibagi dengan mereferensikan ke dokumen yang telah dibuat sebelumnya (SKPL dan DPPL) atau ke dokumen lain yang menjelaskan materi persoalan sistem perangkat lunak yang dibuat (misalnya prosedur kerja, kontrak, dll). Bagian ini harus memberikan:

Daftar lengkap dari dokumen yang direferensikan.

Identifikasi dari setiap dokumen berdasarkan judul, nomor laporan, tanggal dan penerbit

Sumber dari tiap dokumen yang direferensikan.





## 2 Lingkungan Pengujian Perangkat Lunak

Bagian ini akan dibagi menjadi beberapa sub bab, untuk menjelaskan lingkungan yang dibutuhkan dalam pengujian perangkat lunak. Bagian ini juga menjelaskan rencana implementasi dan pengendalian sumber daya (perangkat lunak, perangkat keras dan dari sisi persiapan organisasi) yang akan melakukan pengujian kualifikasi formal.

### 2.1 Perangkat Lunak Pengujian

Bagian ini berisi identifikasi dari nama, nomor dan versi (jika ada atau jika sudah ada), dari item perangkat lunak (misalnya sistem operasi, kompilator, perangkat komunikasi, paket aplikasi yang terkait, basisdata, file masukan, *code auditor*, *Tools* pengujian) yang diperlukan untuk melakukan pengujian. Sebutkan pula hak pemakaian atau lisensi dari tiap perangkat lunak pengujian yang digunakan. Bagian ini juga akan menjelaskan guna dari setiap item, penjelasan media yang digunakan, dukungan peralatan (jika ada) dan masalah keamanan yang berhubungan dengan item perangkat lunak.





## **2.2 Perangkat Keras Pengujian**

Bagian ini berisi identifikasi dari nama, nomor dan versi (jika ada) dari perangkat keras yang dilibatkan dalam pengujian, peralatan khusus (misalnya *interface card* khusus), peralatan komunikasi (jaringan dan peralatannya), dan peralatan lain yang mungkin terlibat.

## **2.3 Material Pengujian**

Beberapa material tambahan yang mungkin dibutuhkan dapat diperjelas dibagian ini. Material ini misalnya manual perangkat lunak, listing program, media yang berisi perangkat lunak yang akan diuji, contoh tampilan keluaran, formulir terkait, atau instruksi-instruksi khusus. Material yang dituliskan di sini adalah material yang belum dituliskan di dokumen-dokumen lainnya.

## **2.4 Sumber Daya Manusia**

Bagian ini menjelaskan jumlah, tingkat keahlian, dan kriteria/prasyarat dari sumber daya manusia yang terlibat dalam pengujian, termasuk saat dibutuhkan (tipe pengujian).





## **2.5 Prosedur Umum Pengujian**

### **2.5.1 Pengenalan dan Latihan**

Bagian ini menjelaskan pengenalan dan latihan yang akan diberikan sebelum dan selama pengujian, bila ada. Informasi yang berhubungan dengan orang yang terlibat sudah dijelaskan di 2.4. Pelatihan ini termasuk instruksi penggunaan perangkat lunak bagi pengguna akhir atau operator, instruksi perawatan perangkat lunak dan instruksi pengendalian perangkat lunak berkelompok. Berikan pula jadwal atau waktu kapan dan seberapa lama pengenalan atau latihan ini dilakukan.

### **2.5.2 Persiapan Awal**

Bagian ini akan dibagi menjadi beberapa sub bab, untuk menjelaskan lingkungan yang dibutuhkan dalam pengujian perangkat lunak. Bagian ini juga menjelaskan rencana implementasi dan pengendalian sumber daya (perangkat lunak, perangkat keras dan dari sisi persiapan organisasi) yang akan melakukan pengujian. Bagian ini dapat dijelaskan secara terpisah untuk tiap kelas atau butir uji bila ada persiapan awal khusus yang perlu dilakukan untuk satu kelas atau satu butir uji. Bagian khusus ini dijelaskan pada deskripsi uji di bawah.







### **2.5.2.1 Persiapan Prosedural**

Bagian ini menyatakan persiapan prosedural (manual) yang perlu dilakukan untuk melakukan pengujian. Contohnya: bila pengujian dilakukan di suatu lingkungan khusus, misalnya di ruang komputer, maka untuk melakukan pengujian ini perlu ada izin masuk khusus, izin penginstallan perangkat lunak yang akan diujikan, pencatatan log-book dan lain-lain.

### **2.5.2.2. Persiapan Perangkat Keras**

Bagian ini akan menjelaskan prosedur yang perlu untuk menyiapkan perangkat keras untuk pengujian. Acuan dapat dibuat untuk menerbitkan petunjuk operasi dari setiap prosedur ini. Pada bagian ini misalnya akan menyatakan hal-hal berikut:

Perangkat keras yang akan digunakan, nama dan nomor jika ada

Setting dari switch (misalnya untuk printer)

Instruksi langkah-langkah untuk penyiapan perangkat keras hingga siap pakai.

Paragraf ini berisi identifikasi dari nama, nomor dan versi (jika ada) dari perangkat keras yang dilibatkan dalam pengujian, peralatan antarmuka (*interface*), peralatan komunikasi, peralatan pengujian waktu (jika diperlukan), dan peralatan lain yang mungkin terlibat.

Contoh:





## Contoh:

Perangkat keras yang perlu disiapkan antara lain:

- 3 perangkat komputer yang masing-masing dilengkapi dengan:
- 1 harddisk dengan kapasitas minimum 500 MB
- 1 color monitor VGA pada perangkat yang sama tempat harddisk berada
- 32 MB RAM
- 1 keyboard
- 1 Floppy drive
- 1 printer Laser Jet yang terhubung ke salah satu perangkat komputer
- 1 Network Hub
- 3 NIC , yang terpasang pada masing-masing komputer, dan kabel UTP yang terhubung ke masing-masing komputer dengan konfigurasi star dan terpusat di Network Hub

*Bila diperlukan suatu konfigurasi yang khusus, dapat dibuat dalam suatu gambar.*



### **2.5.2.3 Persiapan Perangkat Lunak**

Bagian ini akan menjelaskan prosedur atau tata cara yang diperlukan untuk menyiapkan item yang akan diuji, perangkat lunak yang terkait termasuk data untuk pengujian. Informasi yang mungkin perlu ada antara lain:

Perangkat lunak yang diuji (bisa dalam bentuk media penyimpanannya misalnya disket, cdrom, atau media lain)

Perangkat lunak yang digunakan untuk menguji (misalnya simulator, test driver, database)

Instruksi untuk mengaktifkan program, termasuk urutan langkah rincinya bila perlu

Instruksi untuk inisialisasi umum untuk suatu kasus uji.

### **2.5.3 Pelaksanaan**

Bagian ini menjelaskan strategi pelaksanaan pengujian itu sendiri. Contoh strategi ini adalah pembagian pengujian menjadi dua tahap: pengujian unit dan pengujian sistem. Contoh lain adalah: pengujian dilakukan pada lingkungan khusus yang dibangun untuk pengujian dan tidak dilakukan pada lingkungan operasional sesungguhnya.

### **2.5.4 Pelaporan Hasil**

Bagian ini menjelaskan pada siapa saja dokumen hasil pengujian akan diserahkan baik untuk diverifikasi maupun penyerahan akhir.





### 3 Identifikasi dan Rencana Pengujian

Subbagian ini akan dibagi menjadi beberapa sub bagian untuk mengenali kondisi umum pengujian yang dilakukan serta kelas pengujian yang akan dilakukan.

Bagian ini menjelaskan lingkup keseluruhan dari perencanaan pengujian. Dari sejumlah requirement yang akan diuji yang dituliskan di SKPL, buatlah pengelompokkannya dan jadikan tabel pada bagian ini.

Contoh:

Kelas Uji	Butir Uji	Identifikasi		Tingkat Pengujian	Jenis Pengujian	Jadwal
		SKPL	PDHUPL			
Pengujian Antarmuka Pengguna	Pengujian Pewarnaan	SKPL_Xsoft_01	AU_01	Pengujian Sistem	White Box	12/01/2000 – 15/01/2000
	Penataletakan Window	SKPL_Xsoft_02	AU_02	Pengujian Unit	Black Box	15/01/2000 – 17/01/2000
Pembangkitan kode	Pembangkitan Kode Pelanggan	SKPL_Xsoft_03	BK_01	Pengujian Unit	Black Box	18/01/2000 – 19/01/2000
	Kebenaran Data Pelanggan		BK_02	Pengujian Unit	White Box	19/01/2000 – 20/01/2000

Pada contoh di atas daftar requirement yang akan diuji dituliskan di kolom Butir Uji.





## 4 Deskripsi dan Hasil Uji

Bagian ini diuraikan untuk setiap butir uji yang ada. Setiap butir uji yang ada di sana dijelaskan rincian dalam tabel seperti di bawah ini. Hasil yang didapat baru diisi setelah pengujian dilakukan. Bila perlu ada penjelasan khusus untuk mendeskripsikan tiap kelas pengujian, maka tuliskan sebelum merincikan butir ujinya. Bila perlu ada penjelasan khusus untuk mendeskripsikan tiap butir uji, maka tuliskan sebelum tabel.

### 4.1 <Identifikasi kelas pengujian XXXXX>

#### 4.1.1 <Identifikasi butir pengujian YYYYY>

Identifikasi	Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang Didapat	Kesimpulan



#### 4.1.2 <Identifikasi butir pengujian YYYYY>

contoh:

Identifikasi	Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang Didapat	Kesimpulan
BK_02_01	Pengujian hasil pemasukan data pelanggan oleh operator	<ul style="list-style-type: none"> <li>Buka File data pelanggan</li> <li>Cari rekord dengan data modus pemasukan yang diinginkan</li> </ul>	Kode modus pemasukan operator (01)	01<tgl_lahir>001 01<tgl_lahir>002 01<tgl_lahir>003 dst	01<tgl_lahir> <nomor terurut>	01<tgl_lahir><no_loncat	ditolak
BK_02_02	Pengujian hasil pemasukan data pelanggan oleh pelanggan secara on-line	<ul style="list-style-type: none"> <li>Lihat tanggal lahir pelanggan</li> <li>Lihat kode pelanggan</li> <li>Banding kan dengan rumus pembangkitan kode pelanggan</li> </ul>	Kode modus pemasukan on-line (02)	02<tgl_lahir>001 02<tgl_lahir>002 02<tgl_lahir>003 dst	02<tgl_lahir> <nomor terurut>	02<tgl_lahir><no_ terurut>	diterima



Penjelasan setiap kolom yang ada di atas diuraikan dalam penjelasan di bawah ini:

- **Identifikasi**

Bagian ini mengidentifikasi suatu kasus uji dengan suatu kode yang unik. Bagian ini juga akan berisi guna dari pengujian dan penjelasan singkatnya. Subparagraf berikut ini akan memberikan deskripsi rinci dari setiap kasus uji.

- **Deskripsi**

Bagian ini menguraikan deskripsi singkat tentang kasus uji yang dipilih.

- **Masukan**

Bagian ini akan berisi penjelasan tentang masukan pengujian yang diperlukan untuk suatu kasus uji. Hal-hal berikut dapat dimasukkan, jika perlu:

1. Nama, guna dan deskripsi dari setiap masukan (termasuk akurasi dan jangkauan nilainya)
2. Sumber masukan pengujian dan metode yang digunakan untuk memiliki masukan pengujian
3. Apakah masukan untuk pengujian ini adalah masukan yang nyata atau hanya simulasi
4. Waktu atau urutan pemasukan
5. Perilaku dari data masukan yang akan dikendalikan, misalnya
  1. Pengujian suatu item dengan jumlah tipe data dan nilai yang minimum atau secukupnya
  2. Mencoba setiap item dengan suatu rangkaian nilai bertipe data yang benar dan pada setiap pengujian memeriksa efek overflow, underflow dan kondisi-kondisi jelek lainnya
  3. Mencoba setiap item dengan tipe data yang tidak valid dan nilainya untuk setiap masukan data yang tidak pasti
  4. Pengujian ulang jika mungkin







- **Keluaran yang Diharapkan**

Bagian ini memberikan penjelasan setiap hasil harapan uji untuk setiap kasus uji. Baik nilai hasil sementara atau hasil akhir dapat dinyatakan.

- **Kriteria Evaluasi Hasil**

Paragraf yang berikut ini akan mengidentifikasi kriteria yang digunakan untuk mengevaluasi nilai sementara dan nilai akhir untuk setiap kasus uji. Untuk setiap hasil uji, informasi yang dihasilkan antara lain *misalnya*:

1. Keakuratan atau jangkauan nilai keluaran yang masih dapat diterima
2. Jumlah kombinasi minimum atau alternatif kondisi masukan/keluaran yang akan menunjukkan hasil uji yang dapat diterima
3. Durasi maksimum/minimum pengujian, dalam satuan waktu atau jumlah kejadian
4. Jumlah maksimum interrupt, *halt* atau penyebab lain yang membuat sistem berhenti
5. Batas yang masih dapat diijinkan untuk terjadinya kesalahan
6. Kondisi yang menyebabkan hasil uji tidak memuaskan, sehingga perlu dilakukan pengujian ulang
7. Kondisi dimana keluaran akan diinterpretasikan sebagai adanya kesalahan di data uji masukan, pada file data/basisdata, atau pada tata cara uji
8. Indikasi yang masih dapat diijinkan terhadap pengendali, status dan hasil dari pengujian dan kesiapan untuk kasus uji berikutnya (yang mungkin berupa keluaran tambahan)







- **Prosedur Pengujian**

Pada bagian ini akan dijelaskan prosedur uji untuk setiap kasus uji. Prosedur uji ini akan didefinisikan sebagai sekumpulan langkah yang terurut secara sekuensial. Untuk memudahkan dalam dokumentasinya, prosedur uji ini dapat dimasukkan sebagai lampiran pada paragraf ini.

Tingkat kerincian dari cocok adalah level dimana tingkat tersebut berguna untuk menentukan hasil yang diharapkan dan membandingkannya dengan hasil yang nyata. Hal-hal berikut sebaiknya diberikan dalam prosedur uji (jika mungkin):

1. Operasi pengujian operator dan peralatan yang dibutuhkan pada setiap langkah, termasuk, misalnya, perintah-perintah untuk
  - a) Inisialisasi kasus uji dan mencoba masukan pengujian
  - b) Periksa kondisi pengujian
  - c) Melakukan evaluasi tunggal terhadap hasil uji
  - d) Pencatatan Data
  - e) Penghentian suatu kasus uji.
  - f) Jika diperlukan, instruksi pengambilan data
  - g) Modifikasi basisdata/file data
  - h) Ulangi kasus uji jika tidak sukses
  - i) Lakukan alternatif lain yang dibutuhkan oleh kasus uji
  - j) Hentikan kasus uji
2. Hasil harapan uji dan kriteria evaluasi untuk setiap langkah
3. Jika kasus uji digunakan untuk beberapa kebutuhan (*requirement*), berikan identifikasi yang jelas
4. Aksi yang harus dilakukan bila terjadi program berhenti atau ada kesalahan seperti
  - a) Pencatatan data kritis sebagai indikator untuk referensi
  - b) Sistem berhenti pada suatu waktu
  - c) Berhenti (*halt*) pada suatu perangkat lunak pendukung pengujian
  - d) Sekumpulan sistem dan operator yang mencatat suatu hasil uji
5. Prosedur yang digunakan untuk mereduksi dan menganalisa hasil uji untuk mendapatkan kemungkinan-kemungkinan berikut ini:
  - a) Mengetahui apakah keluaran sudah dihasilkan
  - b) Identifikasi media dan lokasi data yang dihasilkan oleh suatu kasus uji
  - c) Evaluasi keluaran sebagai dasar untuk melanjutkan urutan pengujian
  - d) Buatlah keluaran pengujian terhadap keluaran





## 5 Keterunutan Kebutuhan

Bagian ini akan berisi:

1. Keterunutan (*traceability*) dari setiap kasus uji pada PDHUPL ke kebutuhan sistem. Jika suatu kasus uji terdiri dari banyak kebutuhan, keterunutan harus dibuat dari setiap kumpulan prosedur uji hingga kebutuhan yang diuji
2. Keterunutan dari setiap kebutuhan sistem yang dicakup oleh dokumen PDHUPL ini hingga ke kasus ujinya. Untuk pengujian sistem secara keseluruhan, keterunutan dari setiap kebutuhan sistem secara keseluruhan dari dokumen SKPL. Untuk pengujian sistem, keterunutan dari setiap kebutuhan dalam spesifikasi sistem/subsistem. Jika suatu kasus uji digunakan untuk menangani beberapa kebutuhan, maka isi dari keterunutan ini harus dapat menunjukkan suatu langkah pengujian yang khusus untuk menangani setiap kebutuhan.



# Terimakasih



# Pengujian Regresi (Regression Testing)



- Pengujian Regresi melakukan re-eksekusi terhadap beberapa subset test yang sudah dilakukan sebelumnya untuk meyakinkan bahwa perubahan yang baru tidak berakibat pada hasil pengujian sebelumnya
- Jika software diperbaiki, beberapa aspek dalam konfigurasi juga perlu diperbaiki (dokumentasi, program, ataupun data)
- Pengujian regresi akan meyakinkan bahwa perubahan tidak memberikan perilaku yang tidak diinginkan atau tidak mempengaruhi modul-modul sebelumnya.
- Pengujian regresi dapat dilakukan secara manual dengan mengeksekusi ulang semua kasus uji tetapi menggunakan tools diharapkan akan lebih otomatis

— Misalnya penggunaan JUnit (untuk java), NUnit (.NET) atau PUnit (Php).

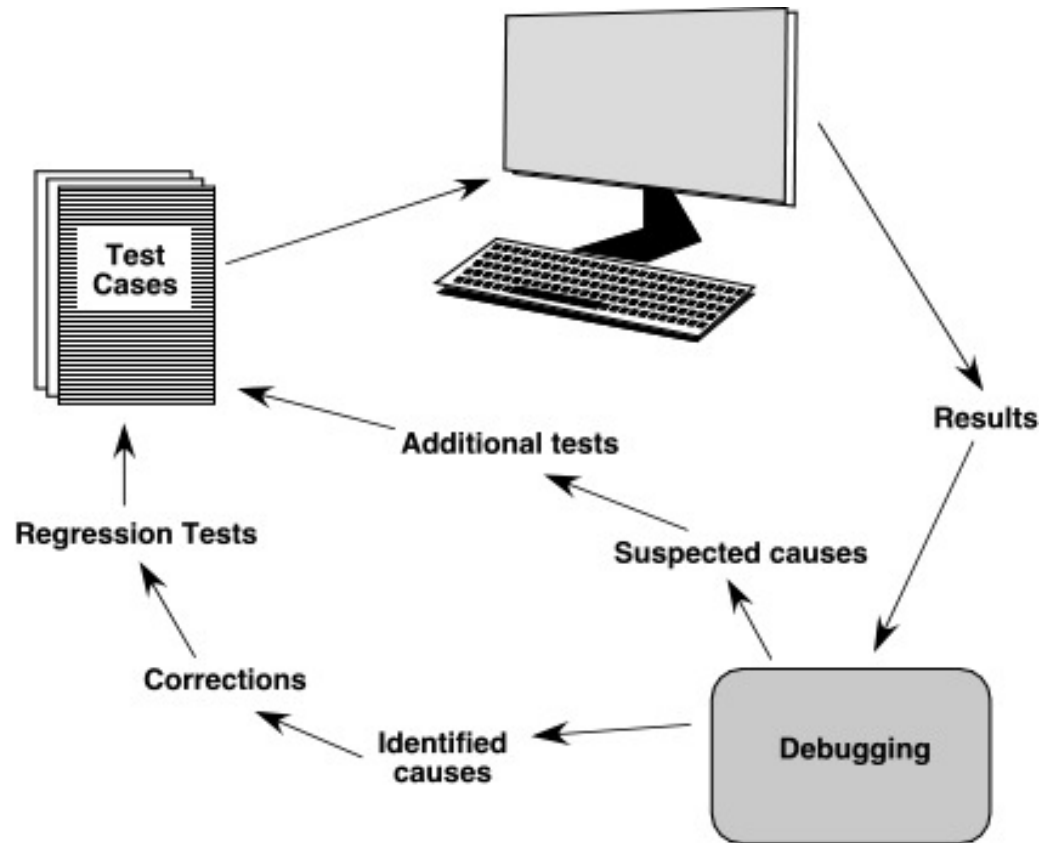




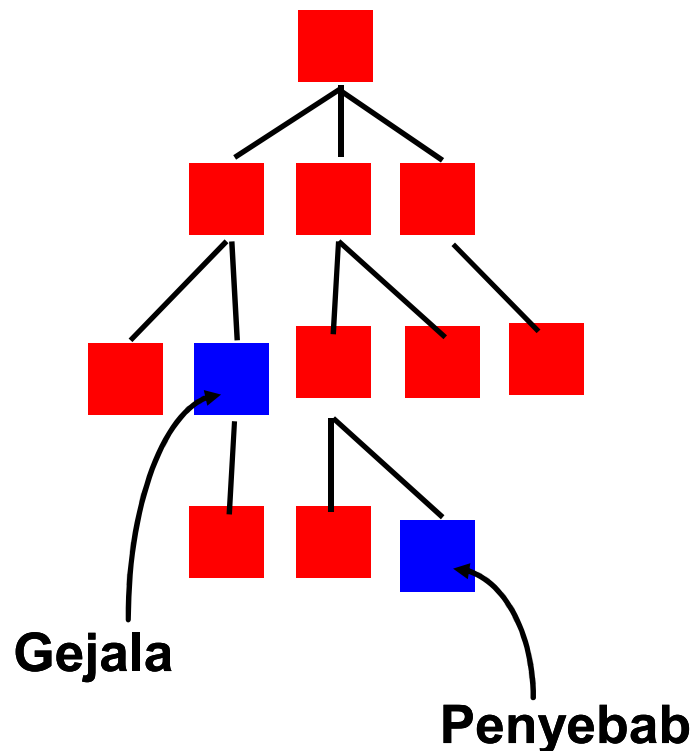
# Debugging: A Diagnostic Process



# Proses Debugging



# Symptoms & Causes



- Gejala dan penyebabnya mungkin terpisah
- Gejala mungkin hilang kalau suatu problem diperbaiki
- Penyebabnya mungkin karena kombinasi yang bukan error
- Penyebabnya mungkin karena sistem/compiler errorr
- Penyebabnya mungkin karena asumsi yang dipercayai benar
- Gejala mungkin muncul mungkin hilang



# Teknik Debugging

- Brute Force/Testing
- Backtracking
- Induction
- Deduction





# Memperbaiki 'Error'



- Apakah penyebab suatu bug bisa direproduksi di bagian program lain
  - Pada banyak situasi, cacat dari suatu program disebabkan oleh adanya suatu pola error yang dari suatu bagian di tempat lain
- Apa bug berikutnya yang mungkin muncul setelah suatu perbaikan (bug fix) dilakukan?
  - Sebelum perbaikan dilakukan maka source code (atau bahkan desain) harus dievaluasi terhadap adanya coupling logik atau struktur data
- Apa yang dapat dilakukan agar suatu bug tidak terjadi sejak awal?
  - Jika 'software proses diperbaiki, maka bug akan otomatis terhindari dari program awal dan juga program-program berikutnya.



# Tugas Kelompok



- Softcopy SKPL-OO dan SKPL-Terstruktur harap dikumpulkan juga via situs kuliah  
→ Rabu, 22 April 2015 (jam 21.00)
- Untuk topik masing-masing (melanjutkan tugas sebelumnya):
  - Buat dokumen perancangan untuk paradigma OO (template: DPPL-OO.docx)
  - Buat dokumen perencanaan pengujian (template: PDHUPL.doc)
- Batas waktu pengumpulan:
  - Jum'at, 1 Mei 2015 jam 21.00 (softcopy via situs kuliah)
  - Senin, 4 Mei 2015 jam 11.00 (hardcopy via Ketua Kelas masing-masing)

