

Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak
Pengujian P/L (Bag 1)

SEMESTER II TAHUN AJARAN 2022/2023



KNOWLEDGE & SOFTWARE ENGINEERING

Kenapa Perlu Pengujian

- Mariner I space probe (1962)
 - Suatu rumus yang ditulis dengan pensil, akhirnya salah diterjemahkan oleh pemrogram
- THERAC-25 radiation machine (1985) :
 - Pengujian kurang lengkap, hingga sempat menyebabkan 3 pasien meninggal
- Rudal Patriot (1991)
 - Salah sasaran karena kesalahan pembulatan (round off error)
- Roket Ariane 5 (1996)
 - Meledak karena kesalahan penanganan konversi komponen 16 bit ke 64 bit
- NASA's Mars lander: September 1999
 - Jatuh karena ada kesalahan dalam integrasi suatu unit
- Millenium Bug (Y2K Bug)
 - Penghilangan digit dua digit awal pada data tahun menjadi masalah ketika pergantian menuju tahun 2000
- Heartbleed (2014)
 - Kerentanan pada OpenSSL
- Video Game 'Pac Man' (2012)
 - Split screen di Level 256



Sumber

- http://en.wikipedia.org/wiki/List_of_software_bugs
- <https://www.computerworld.com/article/2515483/enterprise-applications/epic-failures--11-infamous-software-bugs.html>



Hungarian hacker arrested for pressing F12 (2017)

- The Budapest Transport Authority recently launched an online payment system with the help of a T-Systems Hungary, Deutsche Telekom's consulting arm.
 - The system, which took three months to build The software, not unexpectedly for such a project, was full of bugs including the discovery of an administration screen with a password set to "adminadmin."
- On or about July 14 an 18-year-old – "The boy is nobody. He's not even a programmer," said one Hungarian who wished to remain anonymous – emailed BKK about a hole he found in their system. The hole, if it can be called that, let anyone with passing knowledge of modern browsers to set any price they wanted for any ticket in the system. By simply pressing F12 a "hacker" could change the price of a ticket right in the browser, and because there were no server checks, they could purchase the ticket at that price.



Pengujian Perangkat Lunak

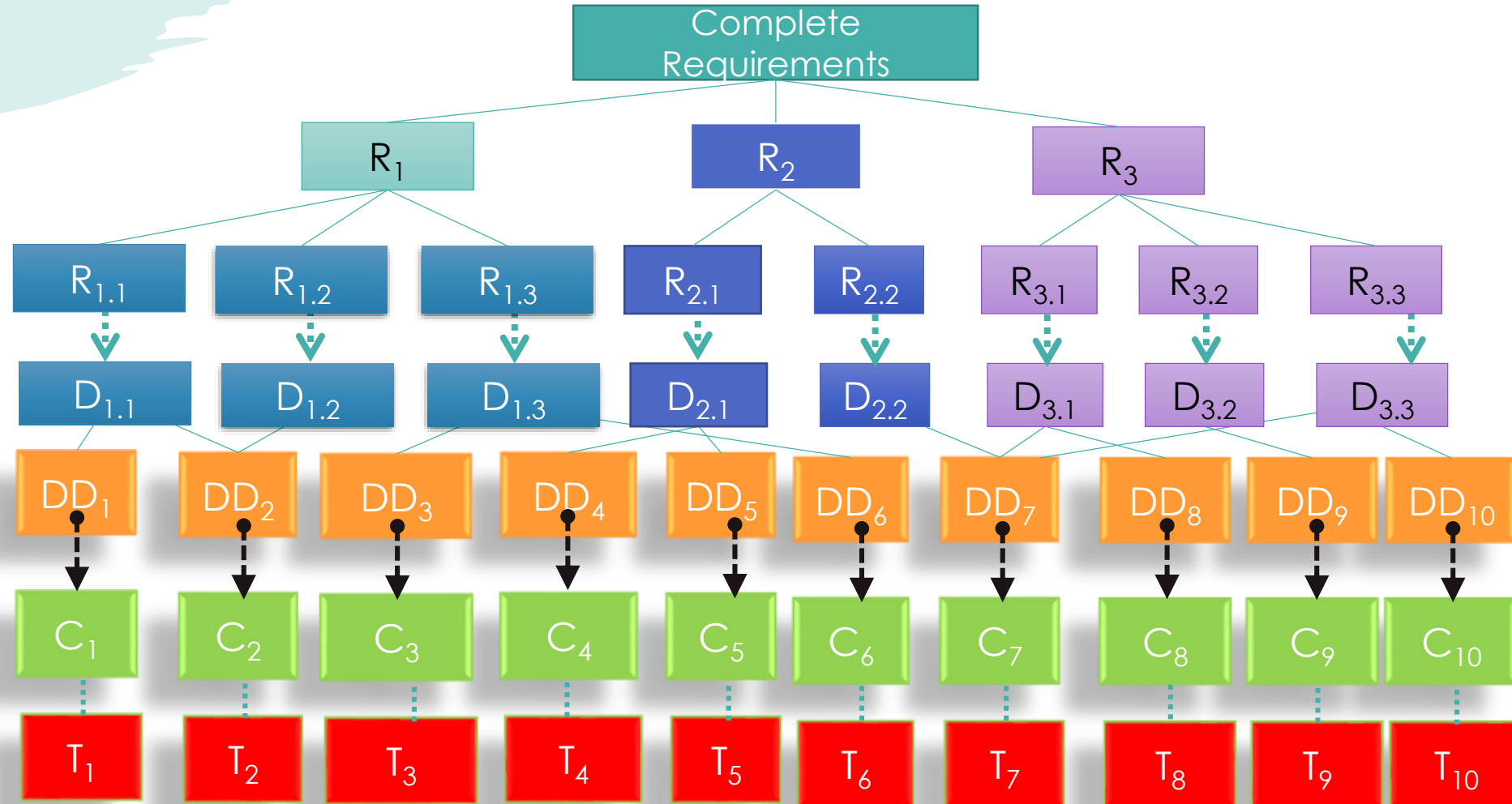
Untuk “Penguji” (bukan pemrogram),
Pengujian adalah proses mencoba
program dengan tujuan mencari
kesalahan sebelum akhirnya diberikan
ke pengguna

Untuk “Pemrogram” yang menguji program,
Pengujian adalah proses mencoba program
dengan tujuan menunjukkan tidak ada salah
pada programnya



Aktivitas Pengujian

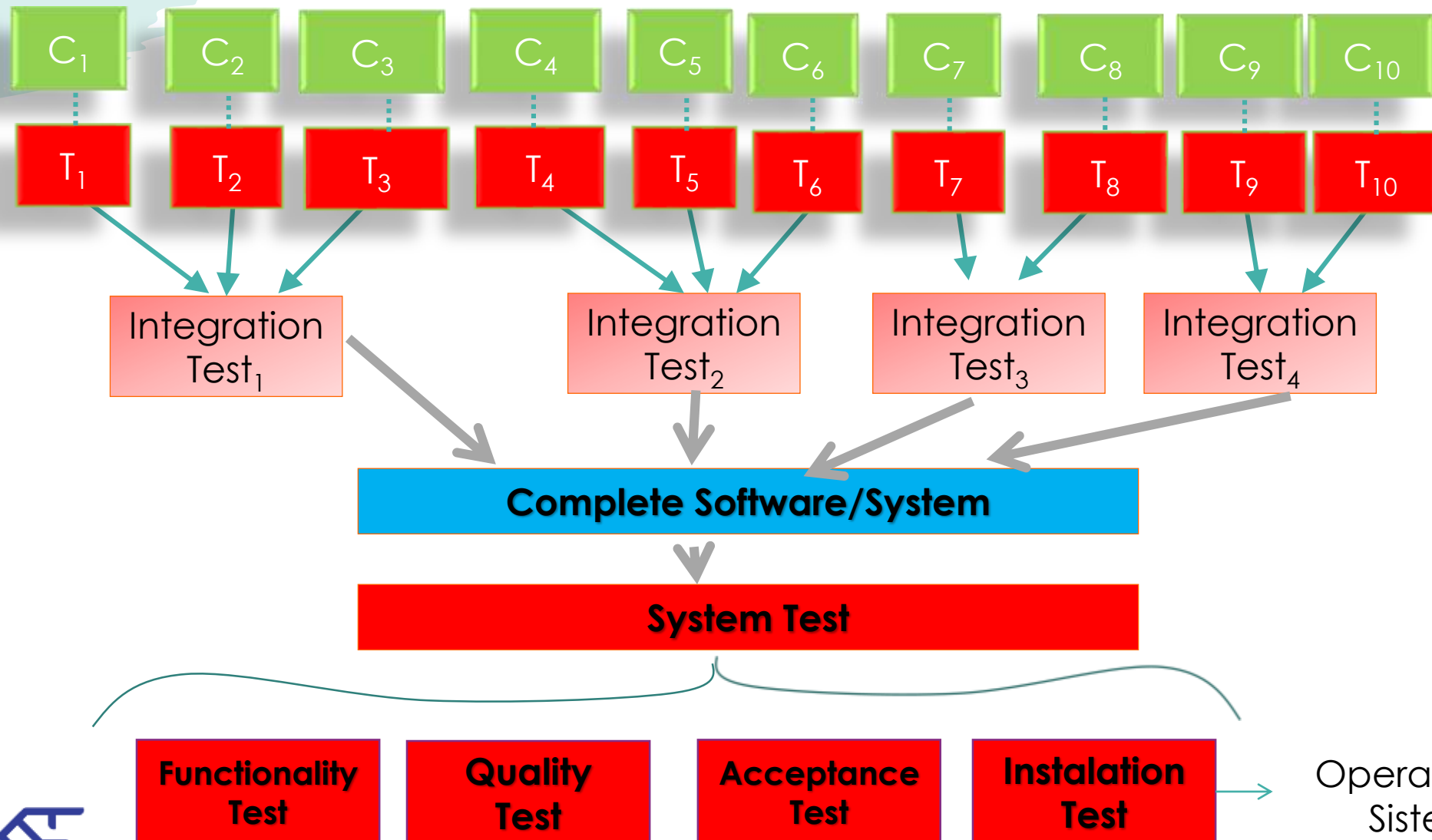
6



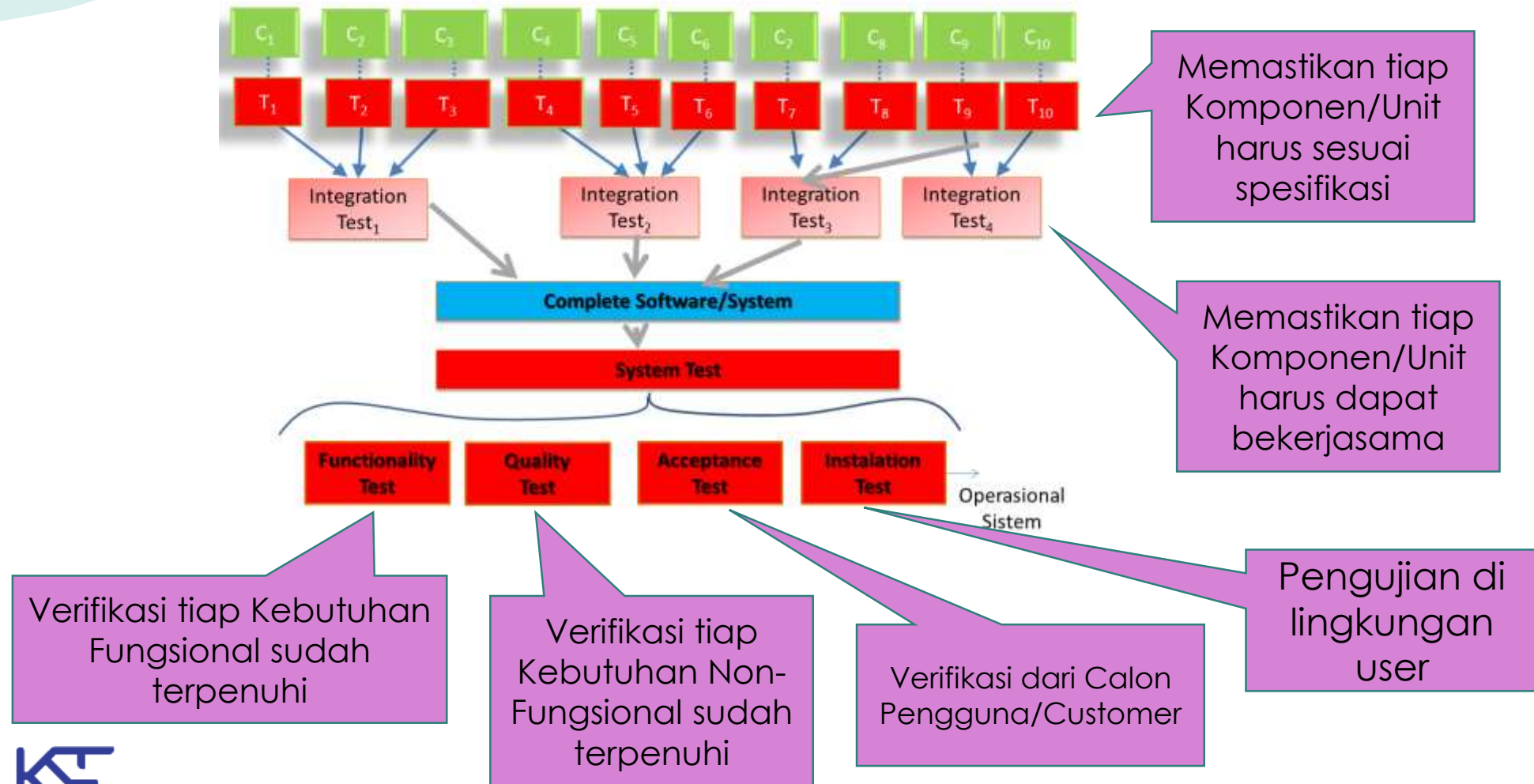
Pemrogram harus melakukan Pengujian terhadap kode programnya

Setiap unit program harus digabung dan hasil penggabungannya diuji kembali (Integration Testing)

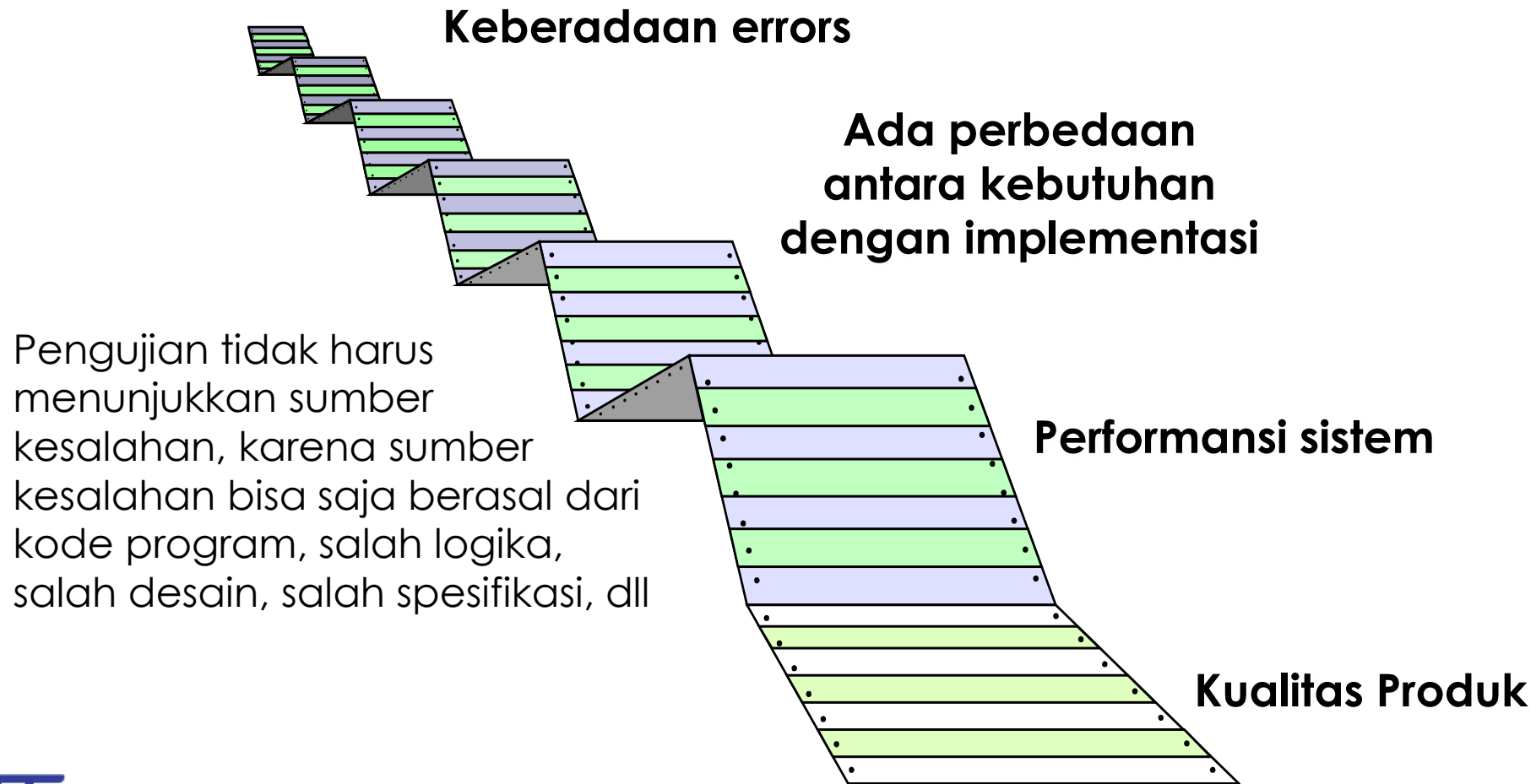
7



Peran tiap pengujian



Pengujian memberikan indikasi:



Failure, Error, Fault and Defect

- **Failure**

- Kegagalan (*failure*) terjadi kalau ada perilaku dari sistem yang tidak sesuai dengan permintaan di spesifikasi sistem

- **Error**

- Error adalah status dari sistem
- Status error ini bisa menyebabkan kegagalan jika tidak ada perbaikan

- **Fault**

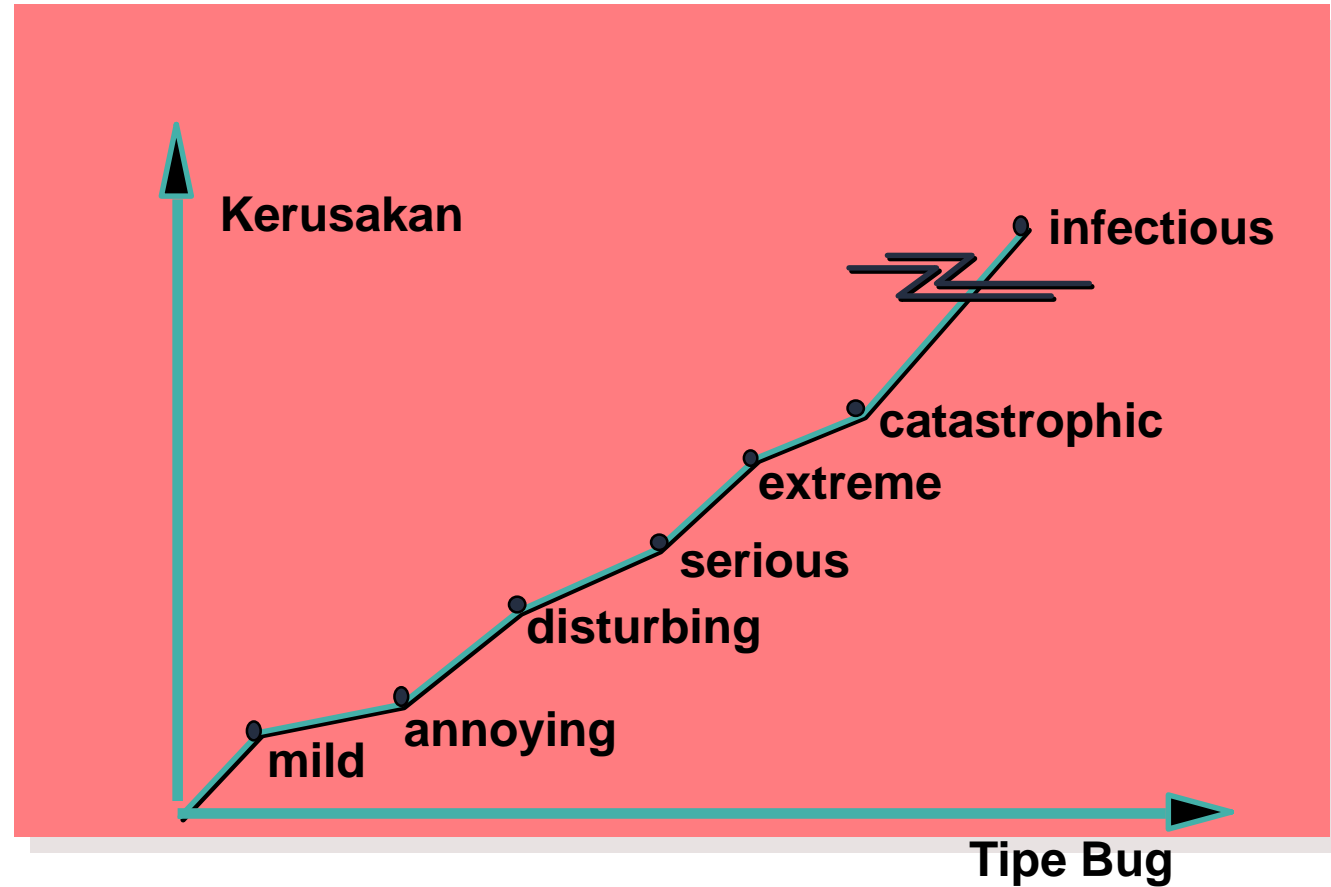
- Fault adalah sumber dari error

- **Defect**

- Sinonim dengan **Fault**
- Sering disebut juga **bug**



Akibat dari Bug



Kategori Bug

- Bug di Fungsi
- Bug di Sistem
- Data Bugs
- Coding Bugs
- Design Bugs
- Documentation Bugs
- Standard Violations

Testing vs. Debugging

- Pengujian (testing) berbeda dengan debugging
 - Kasus uji yang sukses adalah kasus uji yang dapat menunjukkan bahwa ada salah dalam program
 - Kasus uji ini digunakan dalam pengujian
 - Debugging dilakukan bila sudah ditemukan suatu kesalahan, dan tujuannya mencari sumber kesalahan (fault atau defect)
 - 2 Langkah yang dilakukan:
 - Cari kesalahan
 - Perbaiki kesalahan
 - Debugging adalah menjadi bagian dari strategi pengujian



Strategi Debugging (1)

- Tujuan Debugging:
 - Mencari dan membenarkan penyebab terjadinya error
- Bugs dapat ditemukan dengan cara:
 - Evaluasi yang sistematis
 - Intuisi
 - 'Luck'
- Tiga strategi yang biasanya digunakan:
 - Brute Force
 - Backtrack
 - Cause Elimination

Myers, Glenford J., Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.



Strategi Debugging (2)

- **Brute force**

- Teknik ini sering digunakan, dan paling tidak efisien dibandingkan yang lain
- Biasanya digunakan bila cara lain sudah gagal
- Melibatkan:
 - Memory dumps
 - Run-time trace
 - Output Statements
- Sering banyak usaha dan waktu yang terbuang

- **Backtracking**

- Lebih banyak berhasil dilakukan pada program kecil
- *Source code* dibaca dimulai dari lokasi di mana symptom pertama kali ditemukan
- Penelusuran manual dilakukan secara mundur (trace backward) sampai lokasi penyebab error ditemukan
- Pada program besar, jumlah jalur mundur bisa sangat besar, sehingga metode ini menjadi tidak efisien



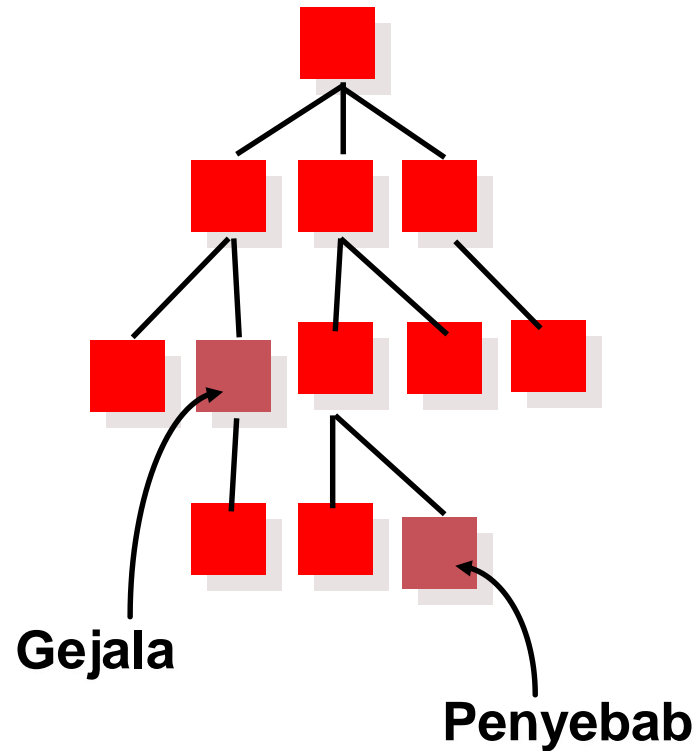
Strategi Debugging (3)

- **Cause elimination**

- Menggunakan **Induksi** dan **Deduksi**
- Induksi – dari detail ke lebih umum
 - Buktikan bahwa suatu kasus spesifik adalah benar, kemudian digunakan untuk membuktikan kasus umum adalah benar
- Deduksi – dari umum ke detail
 - Dari sekumpulan premis umum digunakan sebagai konklusi yang lebih spesifik
- Data yang terkait dengan kemunculan error diatur untuk mengisolasi potensi penyebab terjadinya kesalahan
 - Penyebab terjadinya error dibuat hipotesa, dan data tsb dapat digunakan untuk membuktikan atau sebaliknya
 - Alternatif lain, dibuat sekumpulan daftar penyebab yang mungkin dan pengujian dilakukan berdasarkan daftar tadi
 - Jika salah satu pengujian menunjukkan ada suatu penyebab yang khusus, kumpulan data diperkecil untuk mengisolasi bug

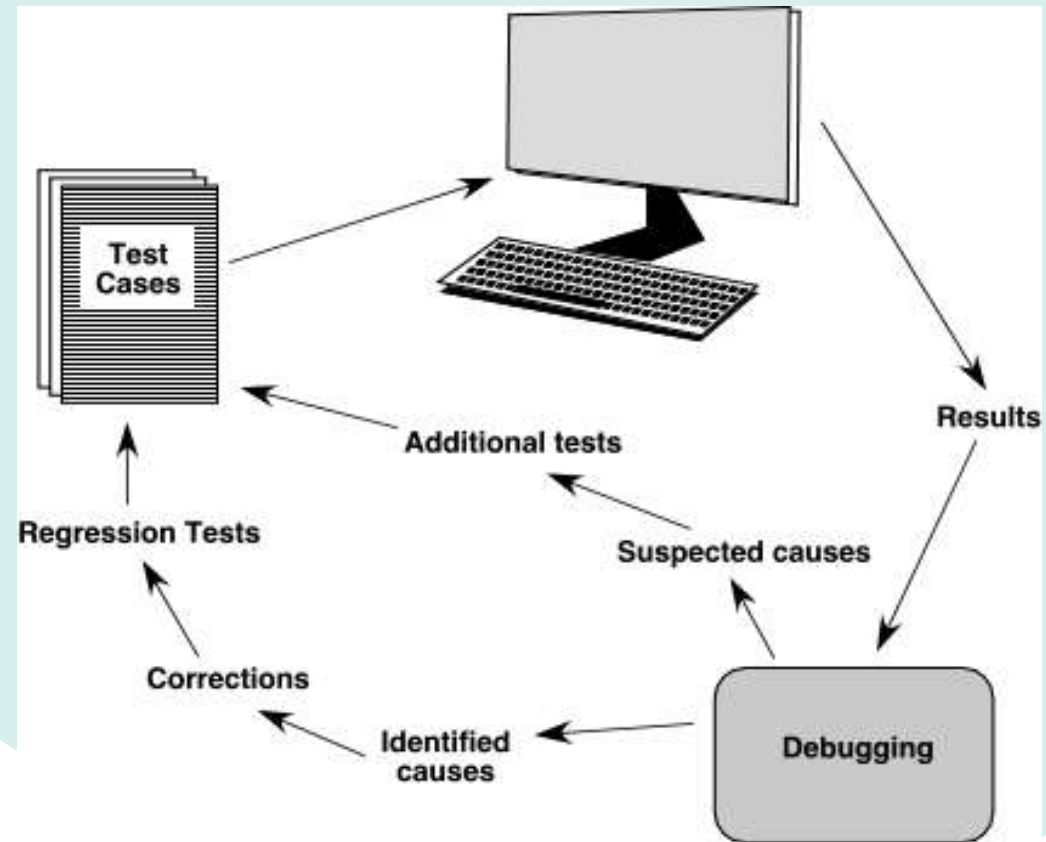


Symptoms & Causes



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

Proses Debugging

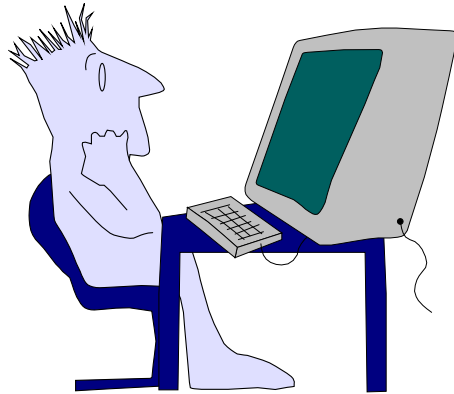


Memperbaiki 'Error'

- Apakah penyebab suatu bug bisa direproduksi di bagian program lain
 - Pada banyak situasi, cacat dari suatu program disebabkan oleh adanya pola error 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.

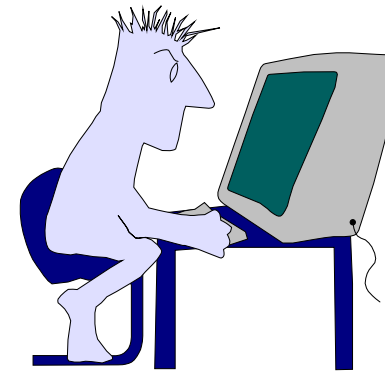


Siapa yang melakukan pengujian?



Pengembang

Mengerti sistem, tapi akan menguji untuk membuktikan bahwa program yang sudah dibuatnya sudah benar



Penguji Independen

Harus belajar sistemnya, tapi akan menguji untuk mencari bahwa program masih ada yang salah

Penguji independen:
Client, User, third party

Aktivitas Pengujian

- Perencanaan Pengujian (Test Plan)
- Perancangan Pengujian (Test Design)
- Eksekusi Pengujian (Test Execution)
- Laporan Pengujian (Test Report)



Aktivitas Perencanaan Pengujian

- Perencanaan Pengujian
- Perancangan Pengujian
- Eksekusi Pengujian
- Laporan Pengujian

- Perencanaan **apa** saja yang akan diuji
- Perencanaan **waktu**
- Perencanaan **sumberdaya** yang diperlukan
 - Orang
 - Perangkat lunak pendukung
 - Alat/perangkat keras
 - Lingkungan pengujian
 - dan lain-lain
- Perencanaan **prosedur umum** pengujian



Aktivitas Perancangan Pengujian

- Perencanaan Pengujian
- Perancangan Pengujian
- Eksekusi Pengujian
- Laporan Pengujian

- Untuk apapun yang akan diuji, maka perlu dirancang **Kasus Pengujian atau Kasus Uji (Test Case)**
- Test Case berisi:
 - **Masukan yang diuji**
 - **Harapan Pengujian**
 - **Cara menilai hasil eksekusi pengujian**



Kasus Uji

Ada masukan data

Ada harapan hasil
(expected
result/outcome)

Masukan	Harapan Hasil
A=0	return 0
A=1	return 1
A=2	return 4

Contoh Pengujian untuk Fungsi Kuadrat (n)

Masukan	Harapan Hasil
n = 0	0
n = 1	1
n = 2	4
n = 3	9
n = 4	16
dst...	

Pertanyaan: Seberapa banyak jumlah kasus uji nya?

23

Kasus Uji yang Baik

- Memiliki kemungkinan tinggi untuk menemukan error
- Tidak redundan (duplikasi yang tidak perlu)
- Pilih teknik yang terbaik
 - Atau tepat sesuai dengan karakteristik software yang diuji
 - Web-based Application vs Mobile-based application
 - Aplikasi Perbankan vs. Aplikasi Perpustakaan
 - Aplikasi Saintifik vs. Aplikasi Sistem Informasi
 - Aplikasi Embedded system vs. Aplikasi Logistik
- Tidak terlalu sederhana juga tidak terlalu kompleks
- Kasus Uji yang sukses adalah jika kasus uji itu dapat membuktikan adanya error!



Aktivitas Eksekusi Pengujian

- Perencanaan Pengujian
- Perancangan Pengujian
- Eksekusi Pengujian
- Laporan Pengujian

- Dilakukan sesudah kode program selesai dikerjakan.
- Eksekusi program dilakukan dengan menggunakan masukan data yang sudah dirancang
- Jika harapan hasil tidak sesuai dengan hasil eksekusi, berarti perlu ada perbaikan program!



Aktivitas Pelaporan Hasil Pengujian

- Perencanaan Pengujian
- Perancangan Pengujian
- Eksekusi Pengujian
- Laporan Pengujian

Laporan pengujian berisi laporan untuk setiap hasil eksekusi Test Case:

- **Lulus pengujian** jika hasil eksekusi program memberikan hasil yang sama dengan harapan hasil
- **Tidak lulus pengujian** jika hasil eksekusi memberikan hasil yang berbeda
- Kadang dapat disertai dengan catatan khusus



Aktivitas Pengujian Pada Daur Hidup Pengembangan PL



Pengujian Kebutuhan Fungsional

**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

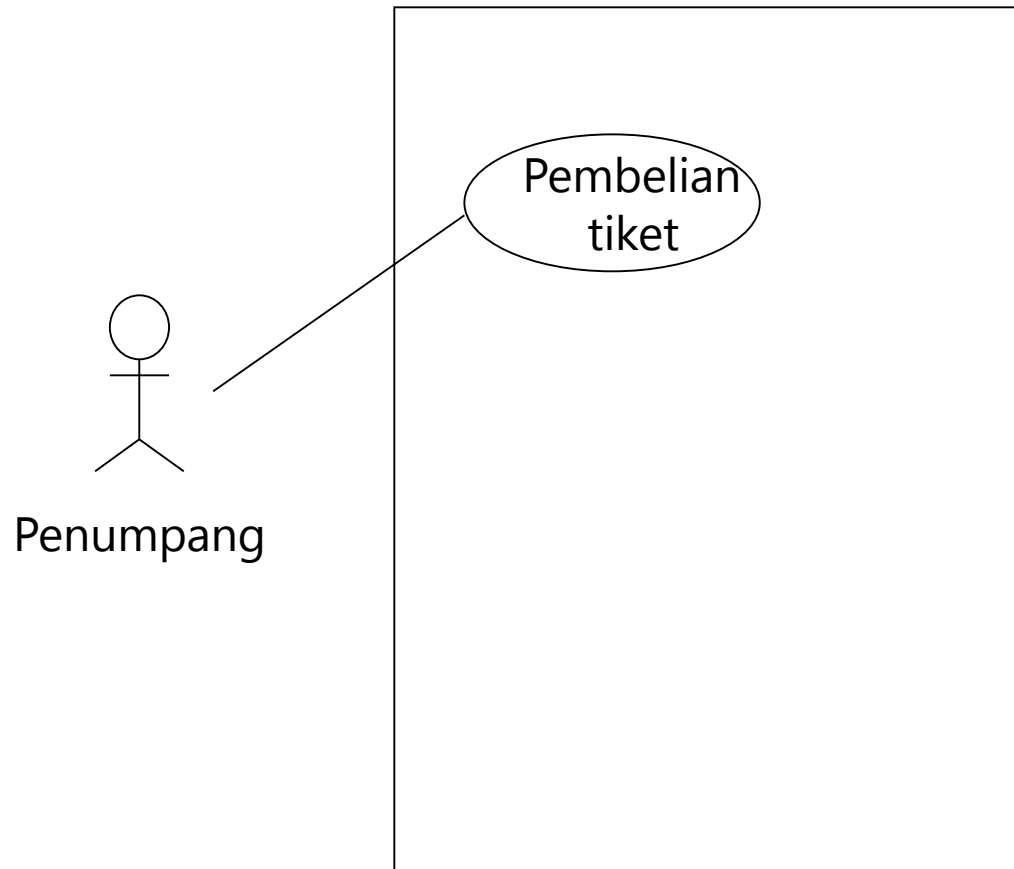


Deskripsi Mesin Tiket Kereta Api

- Sebelum memulai perjalanan kereta api dari suatu stasiun keberangkatan, penumpang harus memilih stasiun tujuan. Mesin akan menampilkan semua tujuan yang mungkin dicapai dari stasiun keberangkatan. Setelah pemilihan tujuan, akan ditampilkan harga tiket.
 - Penumpang bisa mengubah tujuan, dan setiap kali mesin akan memberikan harga yang sesuai dengan tujuan baru.
- Sesudah memilih tujuan, penumpang dengan kategori tertentu bisa mendapatkan discount (misalnya pelajar/mahasiswa, orang tua, memiliki kebutuhan khusus)
- Kemudian penumpang dapat melakukan pembayaran dengan koin dengan nilai tertentu. Mesin akan membaca jumlah koin yang dimasukkan. Jika mencukupi, maka mesin akan mengeluarkan tiket dan mengembalikan uang jika diperlukan.



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">1. Penumpang menekan tombol stasiun 3 dan mesin akan menampilkan 25002. Penumpang memasukkan dua koin 1000 dan satu koin 500
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"> 1. Penumpang menekan tombol stasiun 5, 3, 1, 2, pada saat yang bersamaan setiap kali mesin akan menampilkan 4000, 2500, 1000, lalu 1500 2. Penumpang memasukkan dua koin 1000 3. Mesin harus mengembalikan koin sejumlah 500
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"> 1. Penumpang menekan tombol stasiun 5 dan mesin akan menampilkan 4000 2. Penumpang memasukkan dua koin 1000 3. Lalu penumpang menekan tombol 3 4. Mesin harus langsung mengembalikan koin sejumlah 2000
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">1. Penumpang menekan tombol stasiun 3 dan mesin akan menampilkan 25002. Penumpang memasukkan tiga koin 10003. Mesin harus mengembalikan koin sejumlah 500
Kondisi akhir	Penumpang mendapatkan tiket dan koin kembali sejumlah 500



Pengujian System

- **Validation testing:** fokus pada kebutuhan perangkat lunak (software requirements)
- **System testing:** fokus pada integrasi sistem
- **Alpha/Beta testing:** fokus pada pengguna (bagaimana mereka menggunakan)
- **Recovery testing:** software dibuat 'gagal' dan melihat apakah proses recovery sudah dilakukan.
 - Contoh: software yang memanfaatkan internet dilihat perilaku recovery-nya bila internet tiba-tiba putus
- **Security testing:** memverifikasi apakah mekanisme proteksi sudah melindungi suatu 'penetrasi'
- **Stress testing:** sistem diuji bagaimana jika menghadapi permintaan pemakaian sumberdaya yang melebihi jumlah normal, atau frekuensi atau volume normal
- **Performance Testing:** menguji performansi software saat dieksekusi pada suatu konteks tertentu.



Dokumen Pengujian



Cover

41

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
		GL03-Gxx <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 PDHUP-L-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



KNOWLEDGE & SOFTWARE ENGINEERING

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 PDHUPPL yang dibuat.

1.5 Aturan Penamaan dan Penomoran

Tuliskan aturan penomoran dan penamaan yang dipakai dalam dokumen ini jika ada (misalnya aturan penomoran 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 pengelompokannya dan jadikan tabel pada bagian ini.

Contoh:

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

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 rinciannya 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 YYYYYY>

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"> Buka File data pelanggan Cari rekord dengan data modus pemasukan yang diinginkan 	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"> Lihat tanggal lahir pelanggan Lihat kode pelanggan Banding kan dengan rumus pembangkitan kode pelanggan 	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 berisi 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) Evaluasi 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.

