Perancangan Perangkat Lunak

Pengujian Perangkat Lunak

Avinanta Tarigan

Universitas Gunadarma

Outline

Strategi Pengujian

- Berupa
  - Rencana Pengujian
  - Desain Pengujian
  - Eksekusi Pengujian
- Cukup fleksibel sehingga setiap proyek dapat mempunyai kasus pengujian yang berbeda
- Harus cukup detail sehingga dapat dijadikan tolok ukur kemajuan proyek
- Hasil: Dokumen Spesifikasi Pengujian $\rightarrow$ panduan bagi pelaksana dan milestone bagi manajemen

Karakteristik Pengujian Generik

- Memasukkan "formal technical reviews" untuk mengeliminasi error sebelum pengujian dimulai
- Dimulai dari komponen-komponen sistem sampai pada sistem secara keseluruhan
- Teknik pengujian dipilih sesuai dengan ketepatannya setiap waktu / kasus
- Dilaksanakan oleh pengembang PL atau tim independen
- Pengujian & Debugging tidak sama, tetapi debugging harus ada di setiap testing

Organisasi Pengujian Perangkat Lunak

- Perbedaan konsep Verifikasi (membuat PL dengan benar) dan Validasi (membuat PL yang benar)
- Problem Psikologi:
  - Pengembang cenderung untuk memperlihatkan fitur sistem dan validasinya terhadap kebutuhan user
  - Pelaksanaan oleh tim luar potensi membuat konflik dg pengembang
- Tim luar bukan bertanggungjawab thd kualitas PL

Kapan Pengujian Selesai ?

- Problem: Tidak pernah selesai
- Biaya: Selesai begitu dana untuk Pengujian telah habis
- Kriteria Statistik (Musa & Ackerman)
  - 95% kepercayaan terhadap sistem tsb
  - apabila dalam 1000 jam sisem berjalan terdapat probabilitas 0.995 operasi PL yang tidak gagal

Rencana Pengujian

- Pelacakan Kebutuhan
  - Semua kebutuhan user diuji secara individu
- Item yg diuji
  - Menspesifikasi komponen sistem yang diuji
- Jadual Testing
- Prosedur Pencatatan Hasil dan Prosedur
- Kebutuhan akan Hardware dan Software
- Kendala-kendala
  - Mis: kekuranga staff, alat, waktu dll.
- Proses testing
  - Deskripsi fase-fase utama dalam pengujian

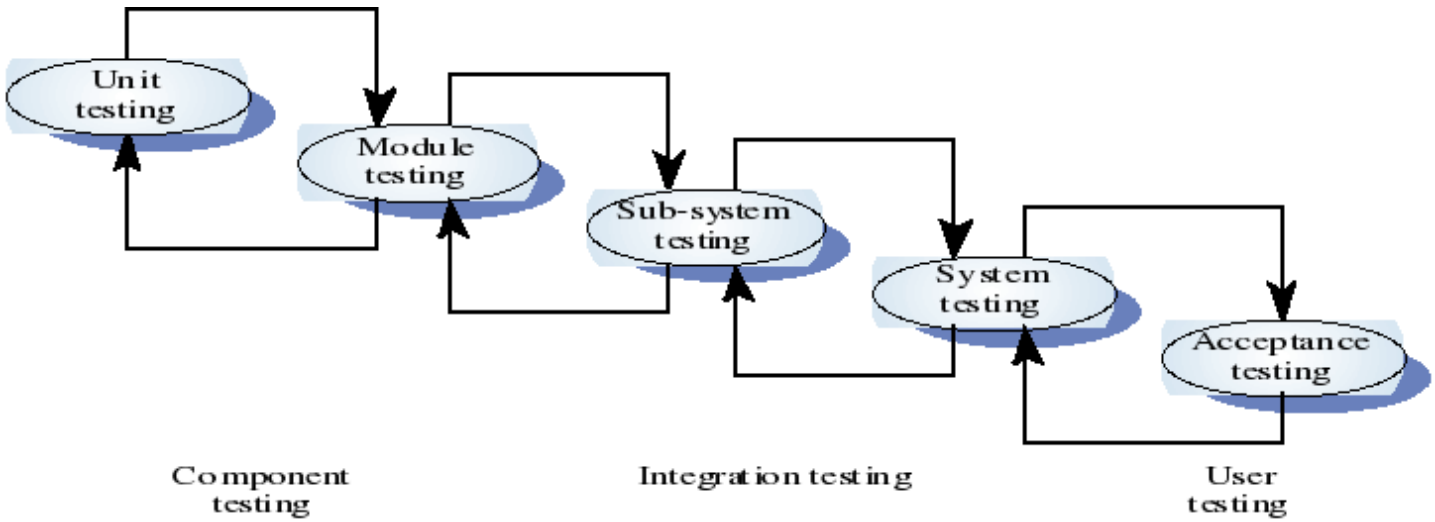
Failure & Faults

- Failure: output yang tidak benar/tidak sesuai ketika sistem dijalankan
- Fault: kesalahan dalam source code yang mungkin menimbulkan failure ketika code yg fault tsb dijalankan

Failure Class	Deskripsi
Transient	Muncul untuk input tertentu
Permanent	Muncul untuk semua input
Recoverable	Sistem dapat memperbaiki secara otomatis
Unrecoverable	Sistem tidak dapat memperbaiki secara otomatis
Non-corrupting	Failure tidak merusak data
Corrupting	Failure yang merusak sistem data

Outline

Proses Pengujian



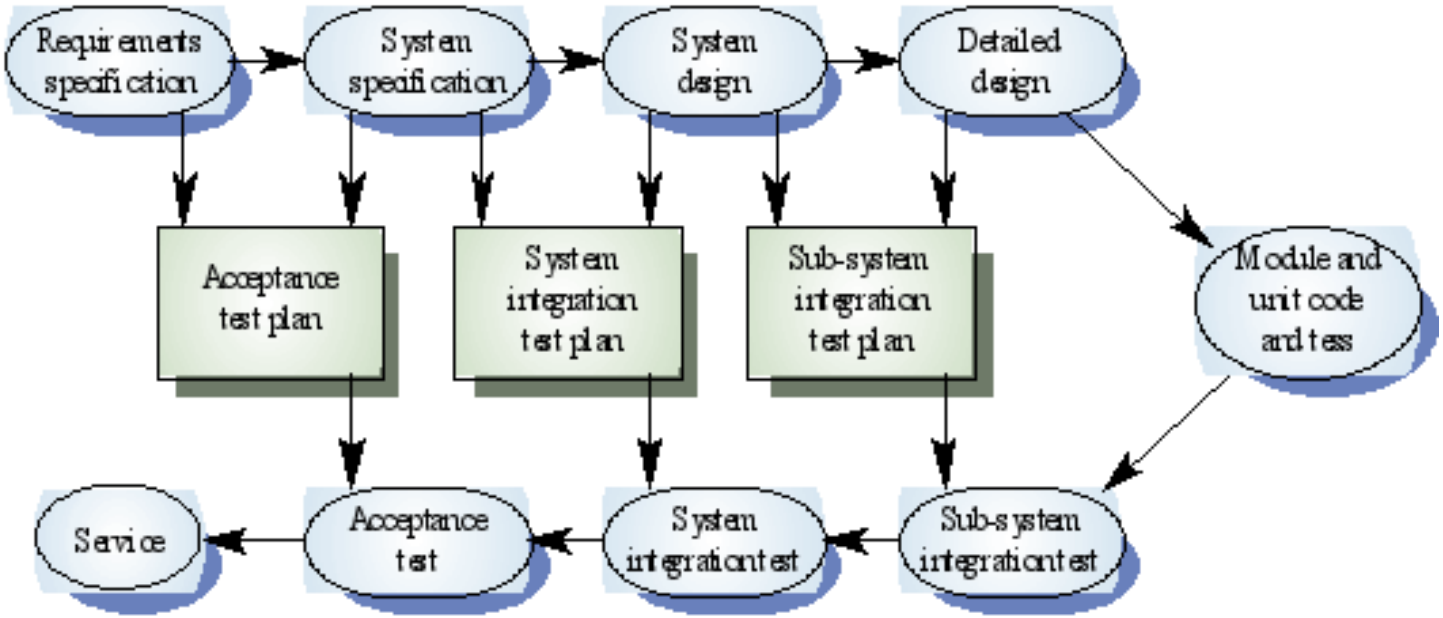
Proses Pengujian I

- Pengujian Unit
  - Komponen-komponen diuji secara individual
  - Pengujian terhadap kode program dan algoritma
- Pengujian Modul
  - Pengujian himpunan komponen-komponen yang saling berkaitan atau bergantungan
- Pengujian Sub-Sistem
  - Pengujian modul yang diintegrasikan kedalam satu sub-sistem.
  - Fokus ada pada pengujian antar-muka
- Pengujian Sistem
  - Pengujian sistem secara keseluruhan
  - Pengujian terhadap adanya "pembrojolan" (emergent properties)

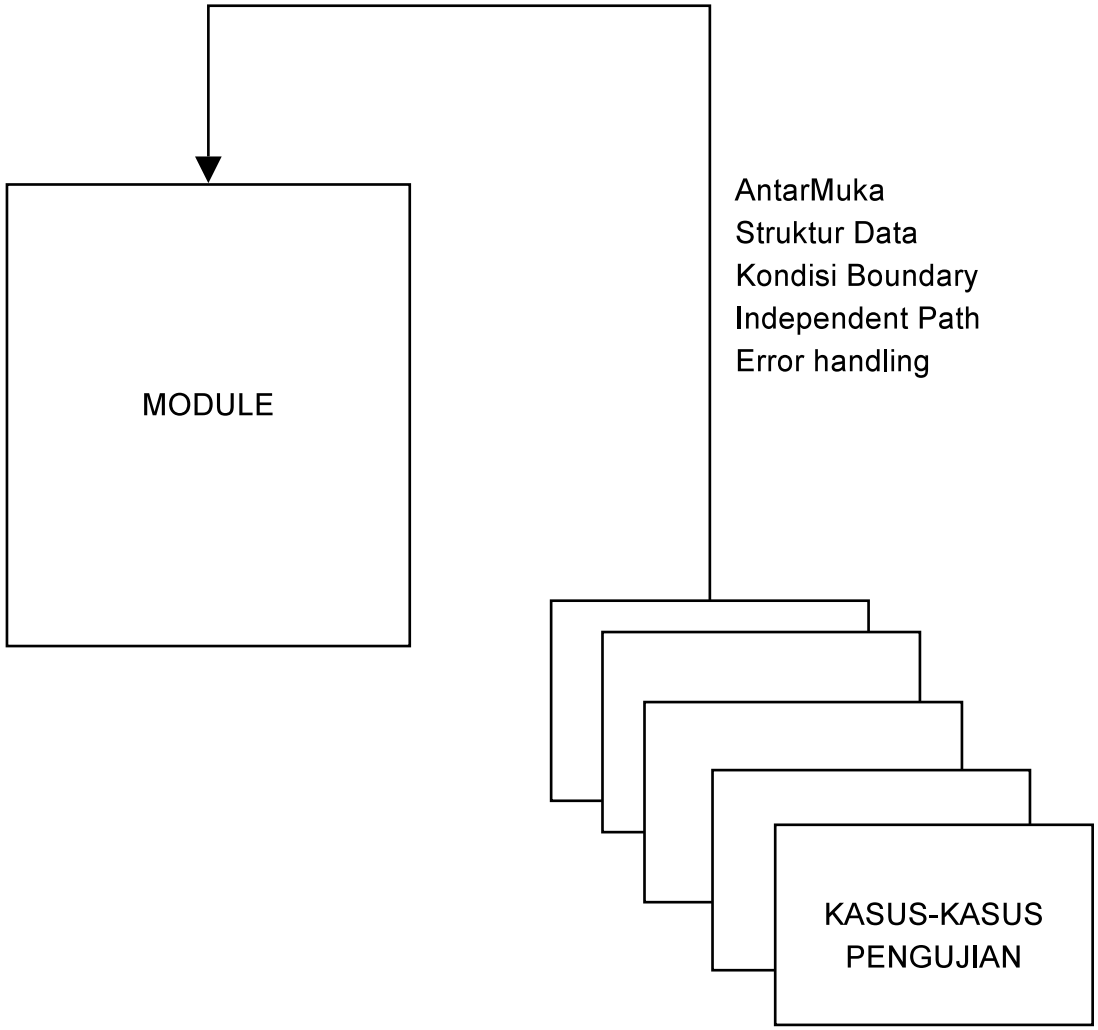
Proses Pengujian II

- Pengujian Penerimaan Pengguna
  - Pengujian Penerimaan Pengguna terhadap PL tersebut
  - Validasi terhadap Kebutuhan Pengguna

Testing Phases



Pengujian Unit



Pengujian Unit

- Antarmuka:
  - untuk memastikan aliran data yang masuk dan keluar sesuai
- Struktur Data Lokal
  - memastikan integritas variabel lokal selama eksekusi
- Kondisi Unit Pada Batas Limit
  - unit selalu beroperasi dengan benar pada limit-limit tertentu
- Independent Path
  - algoritma yang berdiri sendiri beroperasi dengan benar
- Error handling path
  - algoritma untuk mendeteksi dan menangani error beroperasi dengan benar

Path Testing

- Tujuannya meyakinkan bahwa himpunan test case akan menguji setiap path pada suatu program paling sedikit satu kali.
- Titik awal untuk path testing adalah suatu program flow graph yang menunjukkan node-node yang menyatakan program decisions (mis.: if-then-else condition) dan busur menyatakan alur kontrol
- Statements dengan conditions adalah node-node dalam flow graf.

Program Flow Graphs

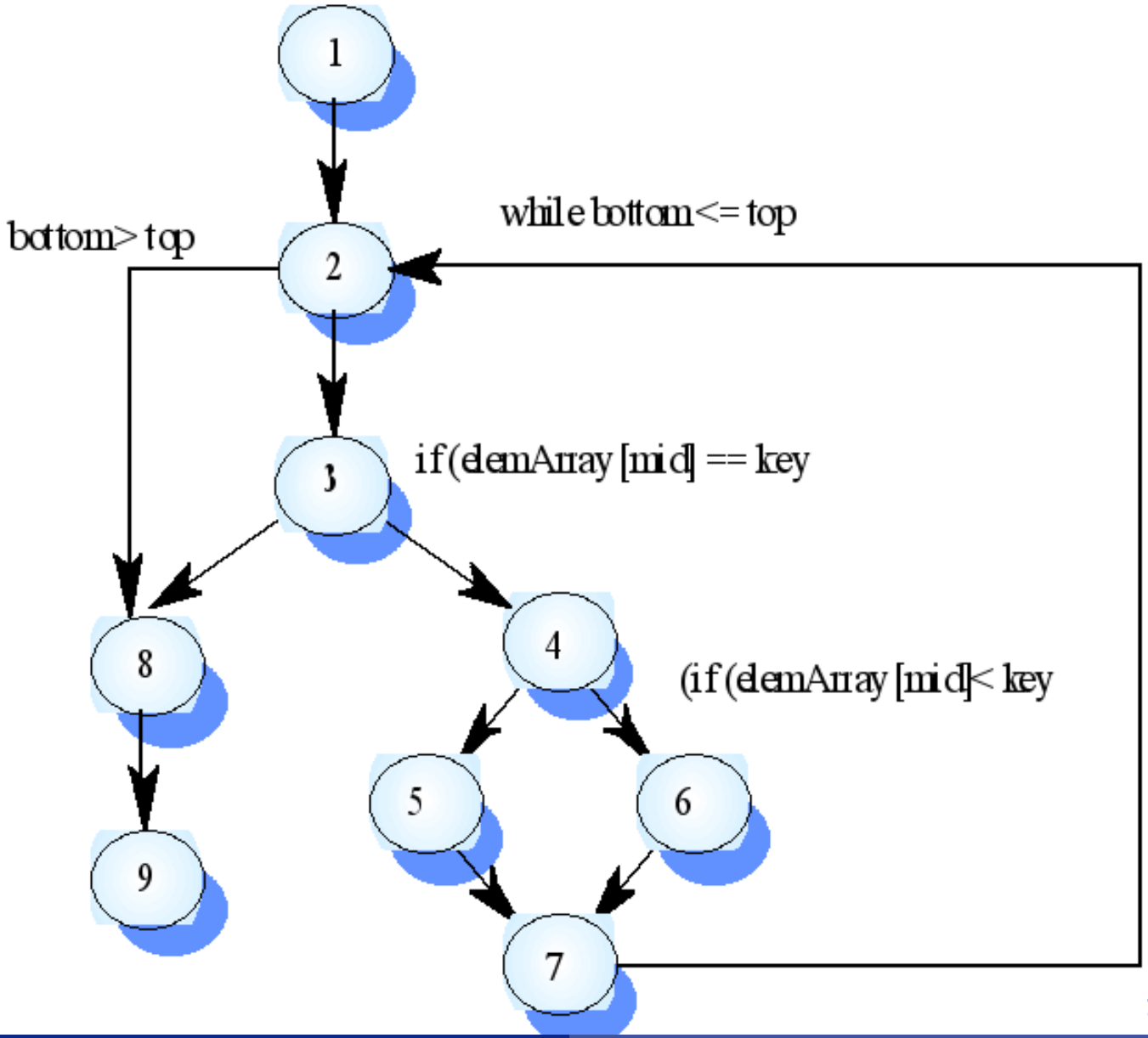
- Menggambarkan alur kontrol. Setiap cabang ditunjukkan oleh path yg terpisah dan loop ditunjukkan oleh arrows looping kembali ke loop kondisi node.
- Digunakan sebagai basis untuk menghitung cyclomatic complexity
- Cyclomatic complexity = Jumlah edges – Jumlah Node + (2 × Number
- Cyclomatic complexity menyatakan jumlah test untuk menguji control statements

Cyclomatic Complexity

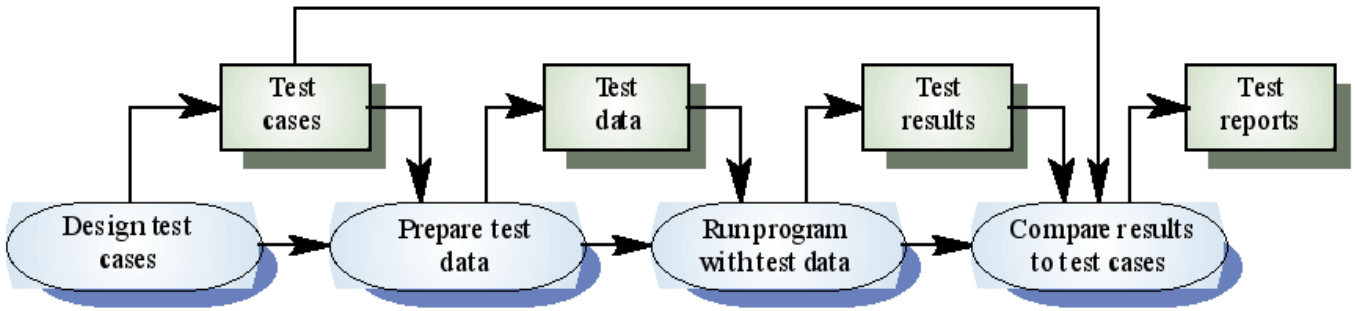
Cyclomatic Complexity	Risk Evaluation
1-10	a simple program, without much risk
11-20	more complex, moderate risk
21-50	complex, high risk program
greater than 50	untestable program (very high risk)

source:http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html

Program Flow Graphs



Defect Testing



Interface Testing

- Parameter interfaces
  - Data dikirim dari satu procedure ke procedure lainnya.
- Shared memory interfaces
  - Block of memory dishare diantara procedure-procedure
- Procedural interfaces
  - Sub-system mengencapsulasi sekumpulan procedure-procedure yang akan dipanggil oleh sub-system lainnya
- Message passing interfaces
  - Sub-systems meminta services dari sub-systems lainnya

Interface Testing

- Interface misuse
  - componen pemanggil memanggil component lainnya dan membuat suatu kesalahan dalam penggunaan interfacenya (mis.: parameter dg urutan yg tidak sesuai).
- Interface misunderstanding
  - component pemanggil salah dalam mengasumsikan behaviour component yg dipanggil.
- Timing errors
  - Component yg memanggil dan yg dipanggil beroperasi pada kecepatan yg berbeda sehingga dimungkinkan mengakses informasi yg tidak uptodate (synchronization problem).

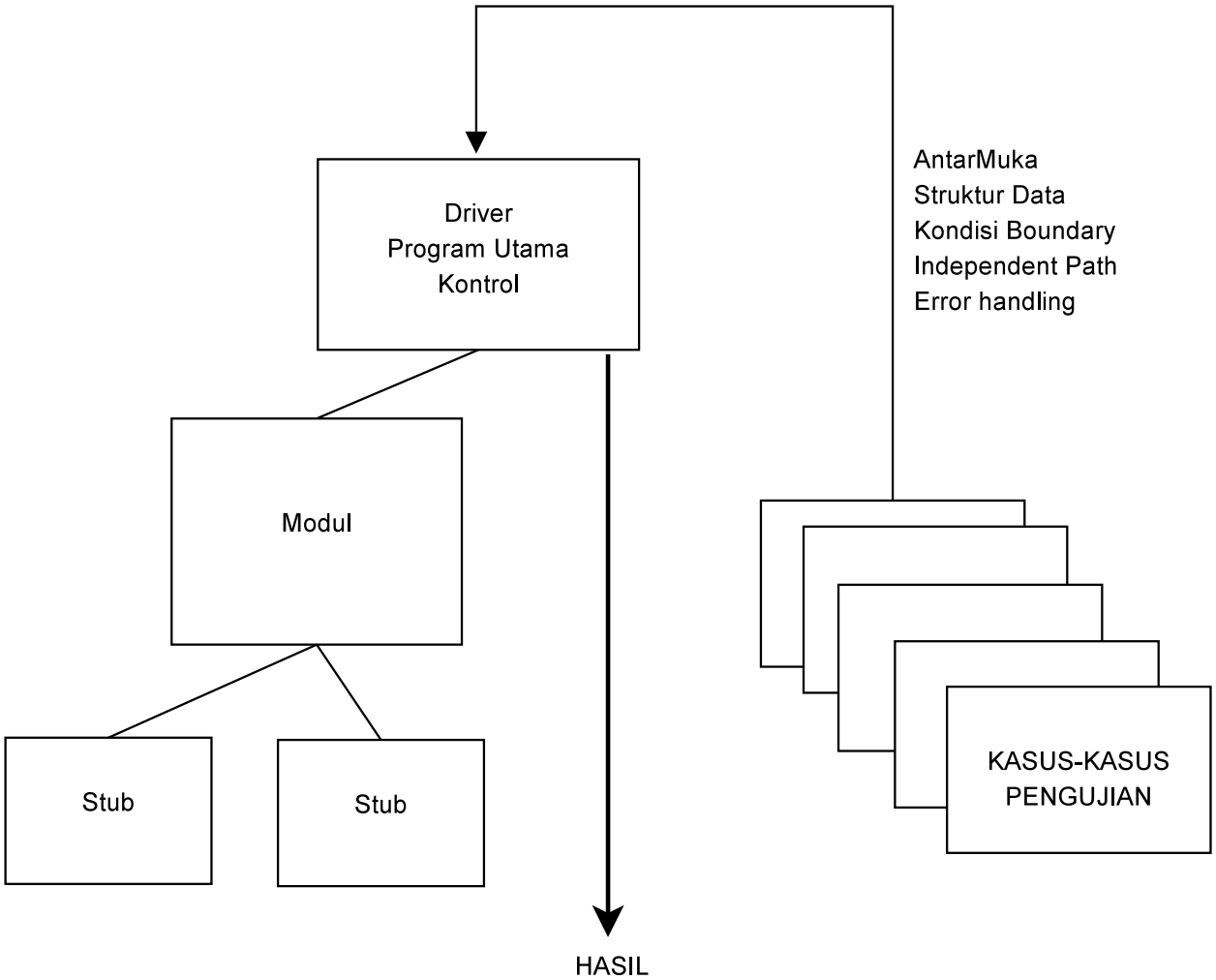
Petunjuk Interface Testing

- Merancang test dimana parameter ke procedure yg dipanggil berada pada nilai batas extrim
- Test Menggunakan null pointer
- Perancangan tests sehingga component yg di test akan fail.
- Menggunakan stress testing pada message passing
- Pada shared memory systems, variasikan urutan dimana komponen diaktifkan.

Kesalahan Umum Dalam Coding

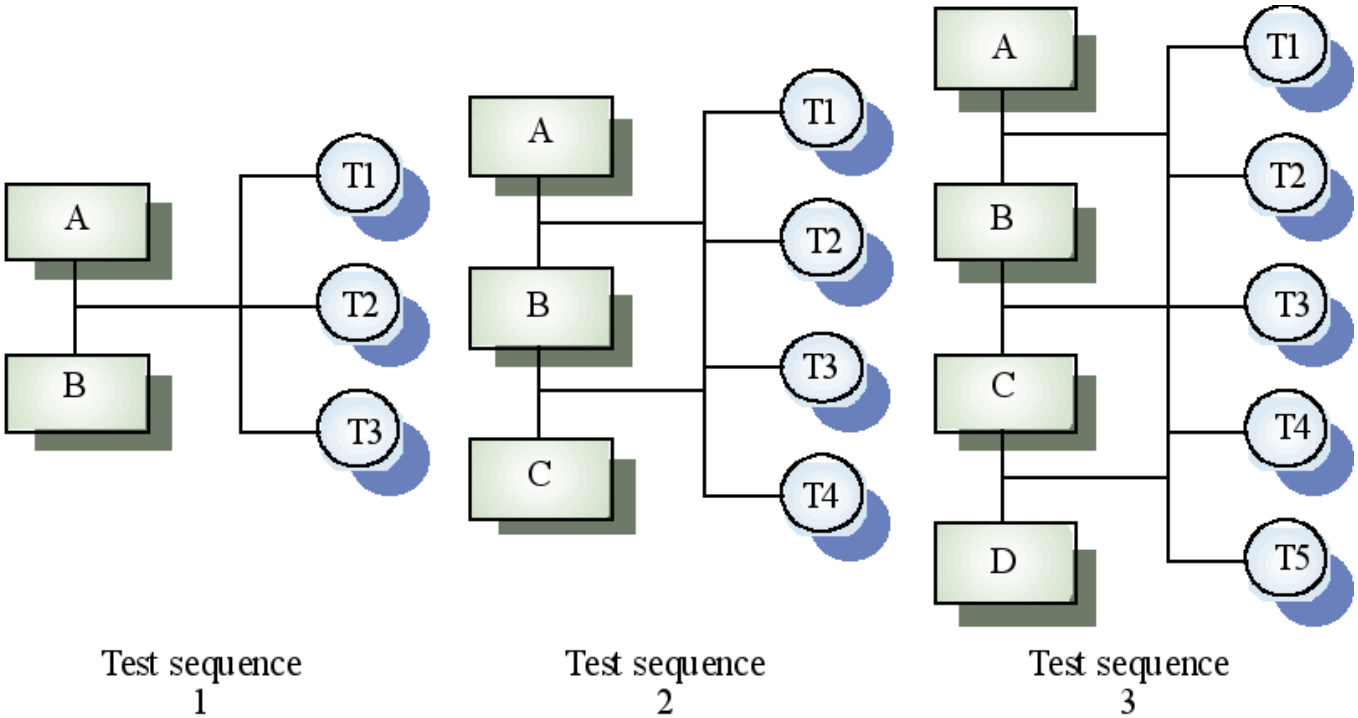
- kesalahan aritmatika
- operasi menggunakan modus yang bercampur
- inisialisasi yang tidak benar
- presisi yang tidak terakurasi
- representasi simbolik yang tidak benar

Pengujian Modul



Pengujian Integrasi Sistem

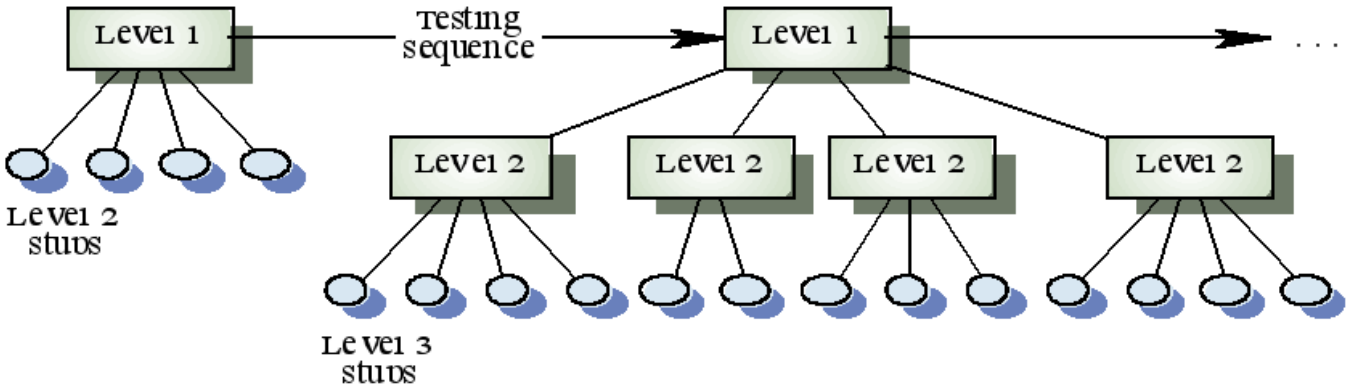
- Incremental Integration vs Big Bang



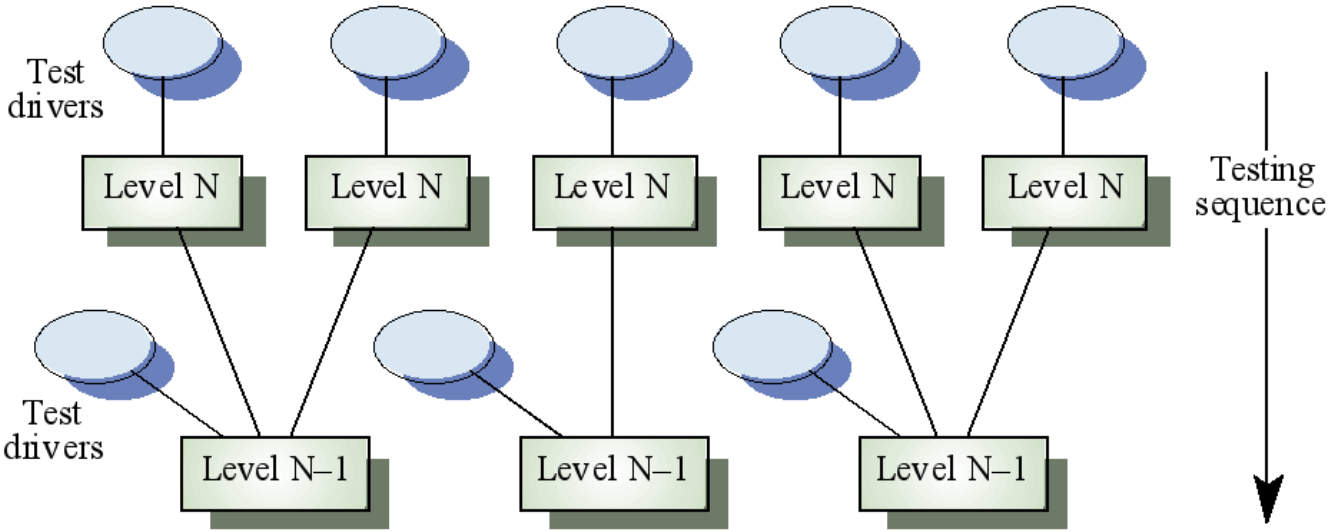
Pengujian Integrasi Sistem

- Top-down Integration
  - Depth First Integration
  - Breadth First Integration
  - Memverifikasi kontrol
- Bottom-up Integration
  - Stubs tidak diperlukan
  - Clustering
- Pada prakteknya, kebanyakan test integrasi menggunakan kombinasi kedua strategi pengujian tsb.

Top Down


Level 1
Testing sequence
Level 1
. . .
Level 2
Level 2
Level 2
Level 2
Level 2 stubs
Level 3 stubs

Bottom-UP


Test drivers
Level N
Level N
Level N
Level N
Level N
Testing sequence
Test drivers
Level N–1
Level N–1
Level N–1

Pengujian Regresi dan Smoke

- Pengujian berulang terhadap komponen / modul yang telah diuji sebelumnya akibat integrasi dengan yang belum diuji
- Smoke:
  - integrasi bertahap dibuat setiap hari dalam bentuk "build"
  - pengujian diulang pada tahap integrasi ini
- Keuntungan
  - Resiko integrasi diminimalisasi
  - Kualitas produk meningkat
  - Memudahkan diagnosa error dan koreksi
  - Kemajuan proyek dapat dilihat dengan mudah

Outline

Pengujian Dalam Konteks OO

- Prinsip enkapsulasi dan information hiding
- Components yang diuji adalah class object yang diinstantiate ke object.
- Lebih besar dibandingkan pengujian sebuah function sehingga pendekatan white-box testing perlu diperluas.
- Tidak jelasnya 'top' suatu system untuk top-down integration dan testing
- Pengujian class-class dan penurunannya
- Detail algoritma dalam setiap class dan keturunannya

Pengujian Integrasi Dalam Konteks OO

- Pengujian Thread
  - pengujian terhadap beberapa class yang tergabung dalam satu thread
  - diuji terhadap input yang ditentukan sebelumnya
- Pengujian Berdasarkan Penggunaan
  - pengujian class terhadap penggunaannya di class yang lain
  - pertama adalah pengujian independent class
  - kedua adalah pengujian dependent class
- Cluster Testing

Testing Levels

- Testing operations pada objects
- Testing object classes
- Testing clusters cooperating objects
- Testing OO system secara lengkap

Object Class Testing

- Complete test yang menguji class melibatkan
  - Testing semua operations suatu object
  - Setting dan interrogating semua attribute object
  - Menguji object untuk semua state (keadaan) yg mungkin
- Inheritance akan mengakibatkan sulitnya perancangan object class tests seperti information yg diuji sulit dilokalisasi.

Integrasi Obyek

- Levels integrasi sedikit berbeda untuk sistem yang berorientasi object.
- Cluster testing digunakan untuk test integrasi and testing clusters terhadap cooperating objects
- Identifikasi clusters menggunakan knowledge dari operation objects dan system features yang diimplementasikan oleh cluster tersebut.

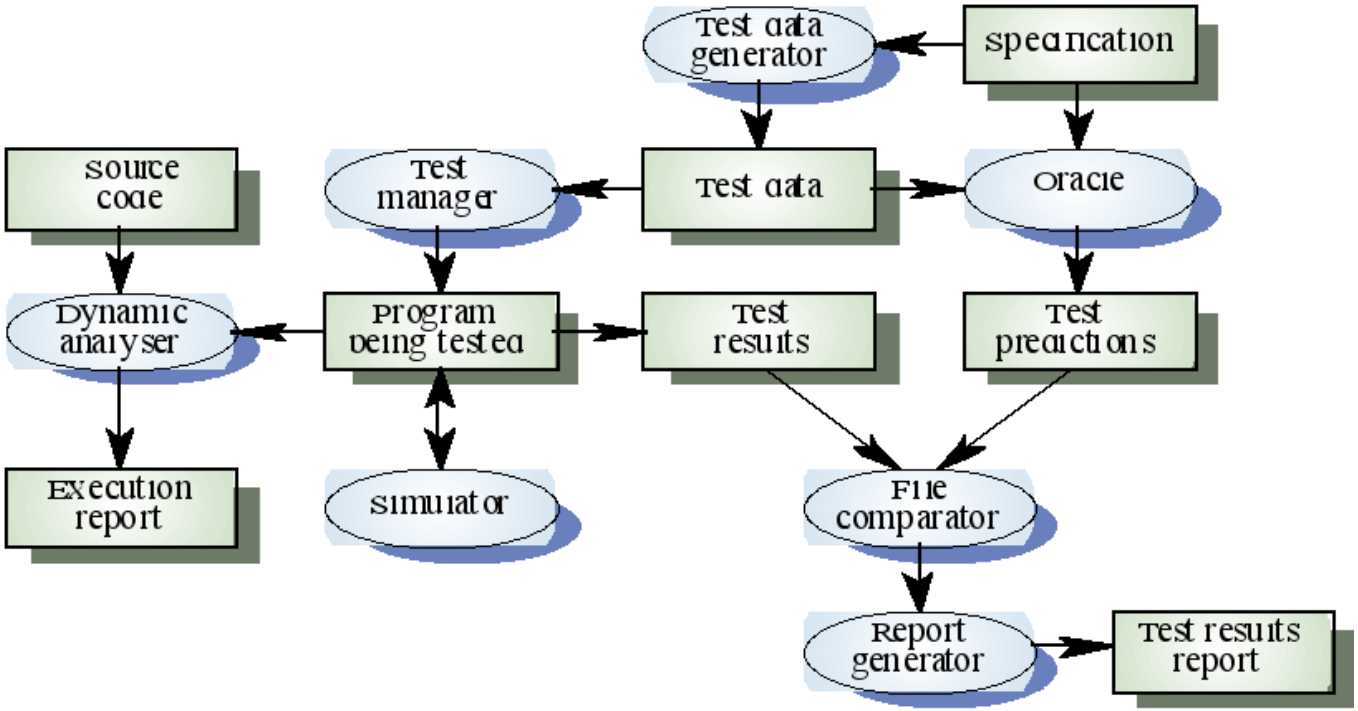
Cluster Testing Approach

- Use-case atau scenario testing
- Testing berdasarkan pada interaksi user dengan sistem.
- Keuntungannya diujikan oleh user yg berpengalaman.
- Object interaction testing
- Tests barisan interaksi object yang berhenti ketika suatu operation object tidak memanggil service dari object lain.

Testing Workbench

- Testing merupakan suatu proses yg cukup mahal. Testing workbenches menyediakan tool-tool untuk mereduksi waktu yg dibutuhkan dan total cost pengujian
- Kebanyakan testing workbenches merupakan open systems karena kebutuhan testing membutuhkan tergantung dr spesifikasi organisasi
- Sulit diintegrasikan dengan closed design dan analysis workbenches

Testing Workbench



Testing Workbench Adaption

- Scripts dibuat untuk user interface simulator dan model test data generator
- Test outputs harus disiapkan secara manual sebagai pembanding.
- Special-purpose file comparators harus dibuat

Pengujian Sistem

- Pengujian Volume
  - Bagaimana sistem dapat terus beroperasi dalam volume pelayanan yang besar
- Pengujian Recovery
  - Bagaimana sistem dapat merekover dirinya thd suatu kesalahan
- Pengujian Keamanan
  - Bagaimana sistem dapat mempertahankan dirinya agar tidak masuk ke dalam "state of insecure"
- Pengujian Stress
  - Bagaimana sistem dapat bertahan beroperasi dalam tekanan "waktu" dan banyak "pelayanan"
- Pengujian Performa
  - Bagaimana performa sistem dalam melaksanakan pekerjaannya

Pengujian Volume

- Menemukan kelemahan sistem selama melakukan pemrosesan data dalam jumlah yang besar dalam periode waktu yang singkat.
- Tujuan: meyakinkan bahwa sistem tetap melakukan pemrosesan data anatar batasan fisik dan batasan logik.
- Contoh:
  - Mengujikan proses antar server dan antar partisi hardisik pd satu server.

Stress Test

- Menguji sistem dengan nilai yg melebihi maksimum load. Stressing suatu system menyebabkan tidak mudah kerusakan.
- Stressing suatu system test failure behaviour. Systems seharusnya tidak gagal total. Stress testing mencek kehilangan service yg tidak diduga ataupun data yg hilang.
- Khusus untuk sistem terdistribusi dapat menyebabkan degradasi jaringan sehingga overload.
- Contoh: Melakukan login ke server ketika sejumlah besar workstation melakukan proses menjalankan perintah sql database.

Performance Testing

- Dilakukan secara paralel dengan Volume dan Stress testing untuk mengetahui unjuk kerja sistem (waktu respon, throughput rate) pada beberapa kondisi proses dan konfigurasi.
- Dilakukan pada semua konfigurasi sistem perangkat keras dan lunak.
- Mis.: pd aplikasi Client-Server diujikan pd kondisi korporate ataupun lingkungan sendiri (LAN vs. WAN, Laptop vs. Desktop)
- Menguji sistem dengan hubungannya sistem ke lain pada server yg sama.
- Load Balancing Monitor
- Network Monitor

Recovery Testing

- Investigasi dampak kehilangan data melalui proses recovery ketika terjadi kegagalan proses.
- Penting dilakukan karena data yg disimpan di server dapat dikonfigurasi dengan berbagai cara.
- Kehilangan Data terjadi akibat kegagalan sistem, hardisk rusak, peghapusan yg tidak sengaja, kecelakaan, virus dan pencuri.

Security Testing

- Privilege access terhadap database diujikan pada beberapa user yang tidak memiliki privilege access ke database.
- Shutdown database engine melalui operating system (dengan beberapa perintah OS) yg dapat mematikan aplikasi database.
- Buffer-Overflow - Menangani parameter input yang melebihi batas buffer
- Penetration-test: password crack, brute-force attack, etc

Kesimpulan

- Pengujian merupakan milestone penting dalam PPL
- Konsep: uji unit yang kecil dan kemudian integrasinya secara bertahap sampai membentuk sistem yang seutuhnya
- Problem dalam pengujian: State Space Explosion Problem
- Pemilihan kasus dan jumlah kasus yang cukup tergantung kompleksitas algoritma dan arsitektur PL
- Banyak kesalahan disebabkan: coding yang tidak sempurna & pengambilan asumsi yang salah

End

Terimakasih