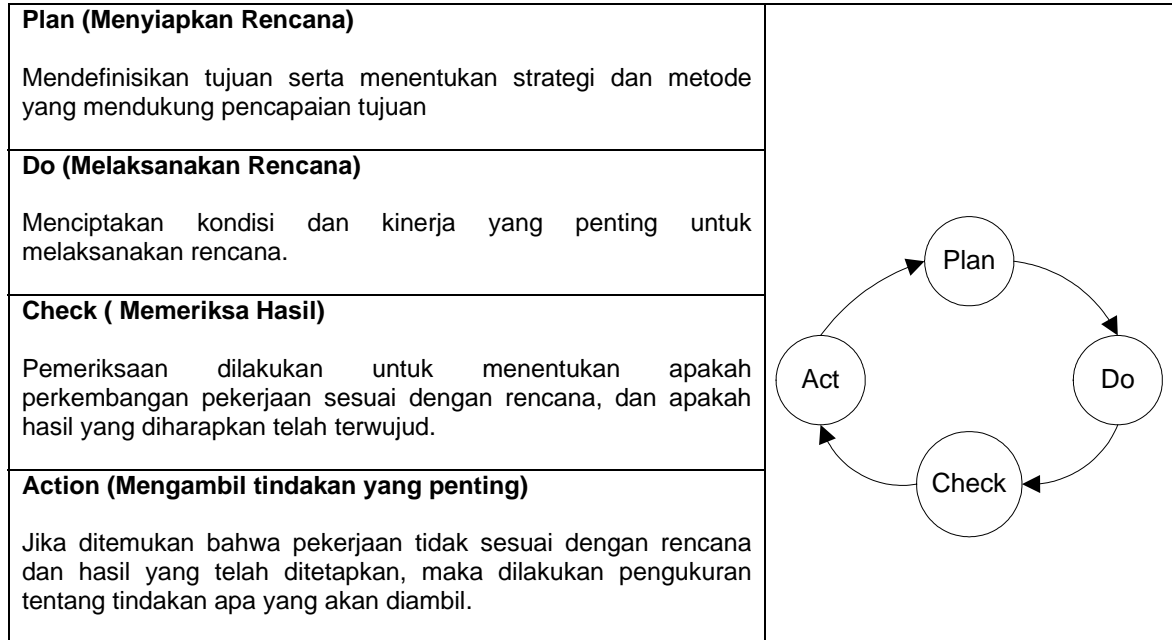

BAB 1 PENDAHULUAN

Secara umum, diketahui bahwa dalam suatu siklus pengembangan perangkat lunak selalu terdapat empat proses utama, yaitu :



Gambar 1 – Siklus Pengembangan secara umum

Penamaan untuk empat proses di atas mungkin akan berbeda, dan pentahapan proses dalam siklus pengembangan juga akan berbeda. Keempat proses tersebut dapat tersebar dalam proses yang ada dalam siklus ataupun dapat berada dalam beberapa proses yang berbeda.

Pelaksanaan kegiatan pada tahap analisis, desain dan implementasi di dalam siklus pembuatan perangkat tidak menjamin bahwa suatu perangkat lunak akan bebas dari kesalahan (*fault free*), untuk mengurangi atau bahkan menghilangkan kesalahan pada perangkat lunak diperlukan suatu tahap pengujian. Kesalahan yang terjadi juga tidak hanya kesalahan yang dapat menyebabkan fungsi perangkat lunak tidak dapat berjalan (*error*), tetapi dapat juga berarti penggunaan perangkat lunak sukar untuk dimengerti user dan proses pelacakan kesalahan sukar untuk dilakukan.

1.1 Definisi Pengujian Secara Umum

Pengujian dapat berarti proses untuk mengecek apakah suatu perangkat lunak yang dihasilkan sudah dapat dijalankan sesuai dengan standar tertentu. Standar yang dijadikan acuan dapat berupa menurut instansi tertentu ataupun disesuaikan dengan keperluan customer/user.

Pengertian pengujian dari masa ke masa :

- 1) Memantapkan kepercayaan bahwa program melakukan apa yang harus dikerjakan.
- 2) Proses mengeksekusi suatu program atau sistem dengan tujuan mencari kesalahan.
- 3) Mendeteksi kesalahan spesifikasi dan penyimpangan dari spesifikasi tersebut.

-
- 4) Semua aktivitas yang ditujukan saat evaluasi suatu atribut atau kemampuan program atau sistem.
 - 5) Pengukuran kualitas Perangkat lunak.
 - 6) Proses mengevaluasi suatu program atau sistem.
 - 7) Memverifikasi bahwa suatu sistem memuaskan atau memenuhi requirement tertentu atau mengidentifikasi perbedaan antara yang diharapkan dengan hasil yang ada
 - 8) Memberitahukan bahwa program melakukan suatu fungsi yang diharapkan secara benar (layak).
 - 9) proses menjalankan dan mengevaluasi sebuah perangkat lunak secara manual maupun otomatis untuk menguji apakah perangkat lunak sudah memenuhi persyaratan atau belum
 - 10) untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya

Berdasarkan definisi di atas, maka dapat disimpulkan bahwa pengujian dilakukan untuk memenuhi persyaratan kualitas perangkat lunak, dengan cara mengeksekusi program untuk mencari kesalahan sintaks program, melakukan verifikasi perangkat lunak untuk melihat kesesuaian antara perangkat lunak dengan keinginan customer.

1.2 Definisi Pengujian Menurut IEEE/ANSI

Definisi menurut IEEE dan ANSI :

- 1) The process of operating a system or component under specified condition, observing or recording the result, and making an evaluation of some aspect of system/component. (IEEE/ANSI, 1990 std 610.12-1990)
 - 2) The process of analyzing software item to detect the difference existing and required condition (that is, bugs) and to evaluate the feature of the software items. (IEEE/ANSI, 1983 std 829-1983)
- (IEEE – Institute of Electrical and Electronics Engineering, ANSI – American National Standards Institute)

Berdasarkan kedua definisi di atas dapat disimpulkan bahwa pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap item perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada setiap komponen system dan mengevaluasi semua fasilitas dari perangkat lunak yang dikembangkan.

Terdapat 2 hal utama yang dilakukan dalam pengujian, yaitu :

- 1) Verifikasi adalah proses mengevaluasi suatu system/component untuk menentukan apakah suatu produk yang diselesaikan setelah fase pengembangan memenuhi kondisi seperti yang telah ditetapkan pada awal pengembangan (saat menentukan spesifikasi) perangkat lunak.
(*"Are we building the product right?"*)
- 2) Validasi adalah proses mengevaluasi suatu system/komponen pada akhir atau selama masa pengembangan untuk menentukan apakah produk yang dihasilkan telah memenuhi kebutuhan-kebutuhan dan persyaratan tertentu yang diminta oleh user.
(*"Are we building the right product?"*).

BAB 2

KETERKAITAN PENGEMBANGAN DENGAN PENGUJIAN

2.1 Tipe Pengembangan Sistem (Proyek Perangkat Lunak)

Tipe proyek pengembangan perangkat lunak harus disesuaikan dengan lingkungan atau metodologi/paradigma yang digunakan dalam pengembangan perangkat lunak.

Tipe Pengembangan Sistem :

A. Pengembangan sistem biasa

- Karakteristik : Menggunakan metodologi pengembangan sistem
User mengetahui requirement
Pengembangan menentukan struktur
- Siasat Pengujian : Pengujian dilakukan pada akhir tiap-tiap tahap
Melakukan verifikasi tiap-tiap spesifikasi yang diperlukan
Menguji struktur dan fungsi

B. Pengembangan Iteratif (Prototyping/CASE)

- Karakteristik : Requirement tidak diketahui
Struktur didefinisikan di awal pengembangan
- Siasat Pengujian : Verifikasi alat Bantu CASE yang digunakan
Verifikasi kebutuhan tiap prototype
Menguji fungsionalitas

C. Pemeliharaan Sistem

- Karakteristik : Memodifikasi struktur
- Siasat Pengujian : Menguji struktur
Mengeluarkan metode kerja yang terbaik
Memerlukan pengujian regresi

D. Kontrak/Pembelian Software

- Karakteristik : Struktur tidak diketahui
Banyak terjadi defect
Fungsionalitas tercantum dalam dokumen
Terdapat berbagai document
- Siasat Pengujian : Verifikasi fungsi yang diperlukan
Pengujian fungsionalitas
Pengujian di dalam lingkungan kerja

2.2 Tipe Pengembangan Perangkat Lunak

Tipe dari system perangkat lunak dapat ditentukan dari proses yang akan dilakukan oleh system. Terdapat 16 tipe system perangkat lunak, dan suatu perangkat lunak dimungkinkan untuk memiliki lebih dari satu system. Sistem-sistem tersebut adalah :

- a) Batch
- b) Event control
- c) Proses control

-
- d) Procedure control
 - e) Advances Mathematical Models
 - f) Messages Processing
 - g) Diagnostic Software
 - h) Sensor and signal processing
 - i) Simulation
 - j) Database management
 - k) Data Acquisition
 - l) Data Presentation
 - m) Decision and Planning Aids
 - n) Pattern and Image Processing
 - o) Computer system software
 - p) Software Development Tools

2.3 Menentukan Lingkup Proyek

Menentukan lingkup suatu proyek ditentukan berdasarkan keseluruhan aktivitas yang tersangkut dalam pembangunan system perangkat lunak. Lingkup proyek menggambarkan karakteristik yang diperlukan secara lebih rinci, dengan menekankan pada daftar requirements yang telah ditentukan terlebih dahulu.

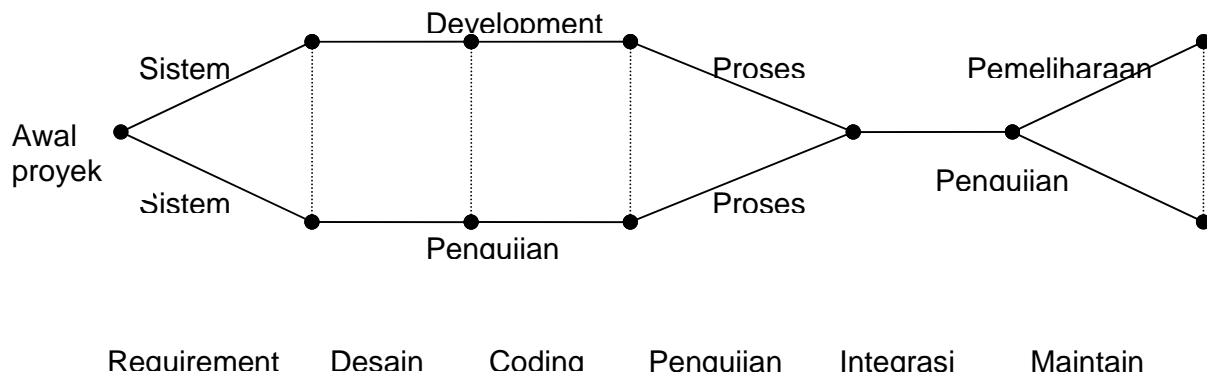
Perbedaan lingkup proyek :

- A. Pengembangan Sistem baru
 - a) Apakah akan mengotomatisasi proses bisnis yang manual
 - b) Proses bisnis yang mana yang akan dipengaruhi oleh system baru
 - c) Area bisnis yang mana yang akan dipengaruhi oleh system baru
 - d) Program Perantara dengan system lama
 - e) Sistem yang lama akan terpengaruh atau tidak
- B. Perubahan system yang telah ada
 - a) Apakah hanya melakukan koreksi
 - b) Apakah hanya melakukan proses rekayasa PL secara standar
 - c) Pengoreksian untuk mengetahui defect dalam rangka perluasan system
 - d) Apakah terdapat system yang terpengaruh
 - e) Adakah resiko atau kemunduran sistem

2.4 Menentukan Waktu Pengujian

Suatu pengujian harus dilakukan selama tahapan proyek.

Konsep siklus hidup pengujian :



Gambar 2 – Siklus Hidup Pengujian

Kegiatan yang dilakukan pada siklus hidup pengujian :

- A. Kegiatan Tahap Requirement
 - a) Menentukan strategi pengujian
 - b) Menentukan kecukupan dari requirement
 - c) Menentukan kondisi pengujian fungsional
- B. Kegiatan Tahap Desain
 - a) Menentukan konsistensi desain berdasarkan requirement
 - b) Menentukan kecukupan desain
 - c) Menentukan kondisi pengujian fungsional dan structural
- C. Kegiatan Tahap Coding
 - a) Menentukan konsistensi desain
 - b) Menentukan kecukupan dari implementasi
 - c) Menentukan kondisi pengujian fungsional dan structural untuk program/unit
- D. Kegiatan Tahap Pengujian
 - a) Menentukan kecukupan dari rencana pengujian
 - b) Pengujian system aplikasi
- E. Kegiatan Tahap Integrasi

Menempatkan pengujian system ke dalam system system keseluruhan.
- F. Kegiatan Tahap Maintenance

Melakukan modifikasi dan melakukan pengujian ulang

BAB 3

PENGUJIAN DALAM SIKLUS PENGEMBANGAN

Pengujian perangkat lunak dilakukan untuk mendapatkan suatu perangkat lunak yang 'layak' untuk digunakan. Suatu perangkat lunak yang telah selesai diujikan harus memiliki standard kualitas tertentu.

3.1 Kualitas Perangkat Lunak

Pengujian dilakukan untuk mendapatkan perangkat lunak dengan kualitas yang baik. Kualitas adalah karakteristik atau atribut dari sesuatu. Kualitas perangkat lunak adalah suatu keadaan yang secara jelas menyatakan permintaan dari fungsi dan kinerja, yang secara eksplisit dituliskan ke dalam dokumen standar pembangunan dan secara implisit menyatakan karakteristik yang diharapkan oleh semua pengembang software.

Pengertian kualitas perangkat lunak terbagi dua tingkat, yaitu kualitas intrinsik produk dan kepuasan customer. Pernyataan pengertian tersebut dinyatakan dalam bentuk pengukuran kualitas perangkat lunak, yaitu :

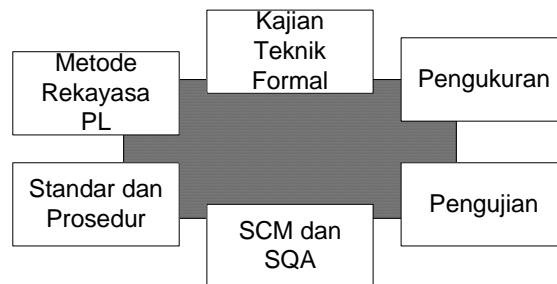
1) Kualitas (intrinsic) produk

Pengukuran dilakukan dengan menggunakan jumlah defect yang terjadi dalam suatu perangkat lunak atau dengan memperkirakan berapa lama perangkat lunak masih dapat berfungsi sebelum terjadi crash.

2) Kepuasan customer

Pengukuran yang dilakukan dengan memperhatikan permasalahan yang dihadapi customer dan tingkat kepuasan customer selama menggunakan perangkat lunak tersebut.

Sedangkan alur pecapaian kualitas yang dapat diharapkan adalah sebagai berikut :



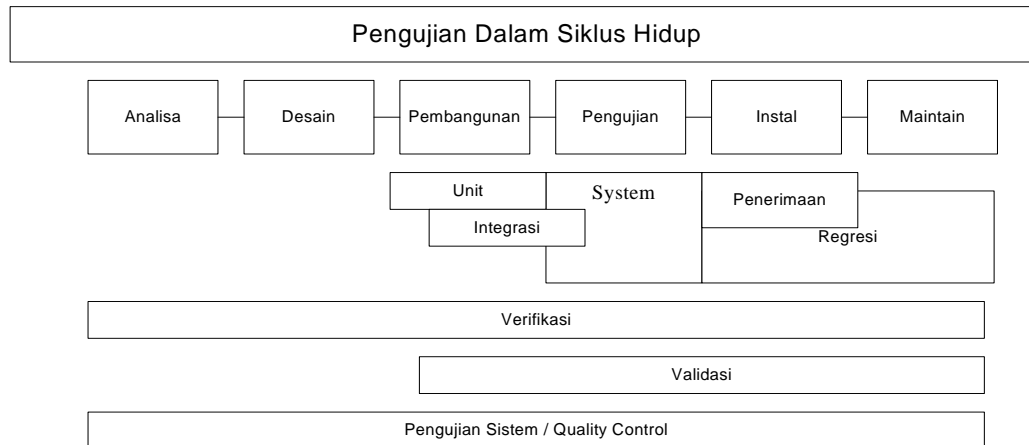
Gambar 3 – Kualifikasi Kualitas

Perancangan perangkat lunak harus mempunyai kualitas:

- Operability,
- Observability,
- Controllability,

- Decomposability,
- Simplicity,
- Stability,
- Understandability,

Di dalam pengujian, terdapat berbagai tipe kriteria pengujian yang dapat digambarkan sebagai berikut :



Gambar 4 – Pengujian dalam Siklus Hidup

Proses Verifikasi dan Validasi adalah keseluruhan proses daur hidup. Verifikasi dan Validasi harus diterapkan pada setiap tahapan dalam proses software. Proses verifikasi dan validasi mempunyai dua obyektif prinsipal, yaitu :

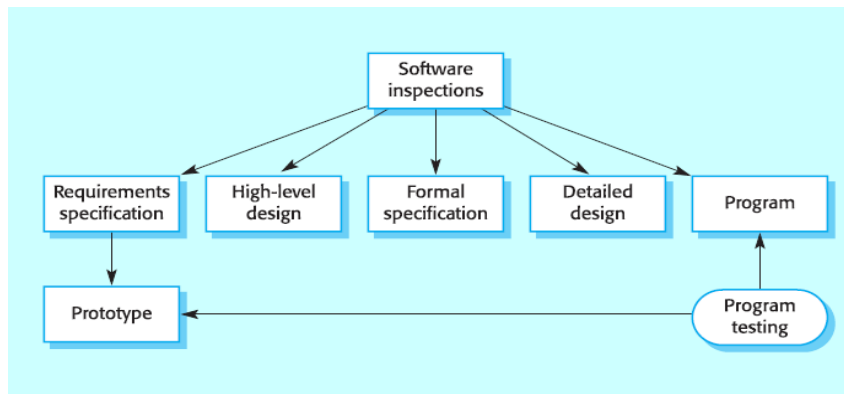
- Menemukan kekurangan dalam sebuah sistem;
- Memperkirakan apakah sistem berguna dan dapat digunakan atau tidak dalam situasi operasional

Tujuan melakukan Verifikasi dan Validasi adalah :

- Verifikasi dan validasi harus memberikan kepastian bahwa software sesuai dengan tujuannya.
- Hal ini bukan berarti benar-benar bebas dari kekurangan
- Harus cukup baik untuk tujuan penggunaannya dan tipe dari penggunaan akan menentukan derajat kepastian yang dibutuhkan.

Terdapat dua kegiatan dalam melakukan verifikasi, yaitu :

- Verifikasi Statik, yaitu berhubungan dengan analisis representasi sistematis untuk menemukan masalah, biasa disebut Software inspection
- Verifikasi Dinamis, yaitu berhubungan dengan dengan pelaksanaan dan memperhatikan perilaku produk, biasa disebut Software testing.



Gambar 5 – Verifikasi Statik dan Dinamik

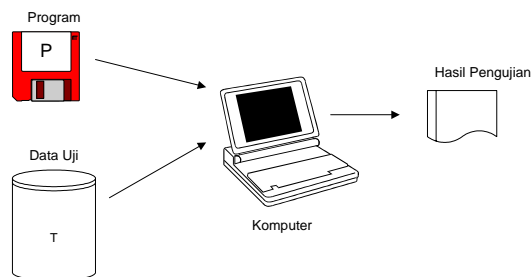
3.2 Sasaran dan Prinsip Pengujian

Glen Myers menyatakan beberapa aturan yang dapat digunakan sebagai penjelasan tentang pengujian perangkat lunak :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan,
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya,
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Maka dapat disimpulkan bahwa pengujian yang baik tidak hanya ditujukan untuk menemukan kesalahan pada perangkat lunak tetapi juga untuk dapat menemukan data uji yang dapat menemukan kesalahan secara lebih teliti.

Secara umum, maka pengujian perangkat lunak digambarkan sebagai berikut:



Gambar 6 – Pelaksanaan Pengujian

Pengujian dilakukan untuk menemukan ketidaksesuaian dengan permintaan costumer/user. Beberapa prinsip dalam pengujian yang dapat dgunakan seperti yang diungkapkan oleh Alan Davis adalah sebagai berikut :

1. Semua pengujian harus dapat ditelusuri sampai ke persyaratan user, yaitu dengan mengungkapkan kesalahan dari cacat yang menyebabkan program gagal.
2. Pengujian harus direncanakan sebelum proses pengujian itu dilakukan, yaitu dengan merencanakan dan merancang semua pengujian sebelum coding dijalankan.

-
3. Prinsip *Pareto* berlaku untuk pengujian perangkat lunak, hal ini berdasarkan pengamatan yang menyatakan bahwa dari 80% kesalahan yang ditemukan selama pengujian dapat ditelusuri sampai 20% dari semua modul program.
 4. Pengujian harus mulai dari yang kecil ke pengujian yang lebih besar dan kompleks.
 5. Pengujian tidak dilakukan secara mendalam dan detail.
 6. Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang independent.

Tujuan pengujian PL :

- Menilai apakah perangkat lunak yang dikembangkan telah memenuhi kebutuhan pemakai
- Menilai apakah tahap pengembangan perangkat lunak telah sesuai dengan metodologi yang digunakan
- Membuat dokumentasi hasil pengujian yang menginformasikan kesesuaian perangkat lunak yang diuji dengan spesifikasi yang telah ditentukan

Melihat tujuan dari pengujian perangkat lunak, maka dapat dijabarkan hal-hal yang harus dilakukan ketika melakukan pengujian, yaitu :

- Mengidentifikasi dan menemukan beberapa kesalahan yang mungkin ada dalam Perangkat Lunak yang diuji
- Setelah Perangkat Lunak dibetulkan, diidentifikasi ulang kesalahan dan dites ulang untuk menjamin kualitas level penerimaan
- Membentuk tes yang efisien dan efektif dengan anggaran dan jadwal yang terbatas
- Mengumpulkan daftar kesalahan untuk digunakan dalam daftar pencegahan kesalahan (tindakan *corrective* dan *preventive*)

3.3 Testabilitas

Di dalam melakukan pengujian terdapat beberapa karakteristik yang harus diperhatikan. Karakteristik pengujian tersebut adalah :

1. Pengujian yang baik memiliki probabilitas yang tinggi untuk menemukan kesalahan
2. Pengujian yang baik tidak redundan.
3. Pengujian yang baik seharusnya jenis terbaik, yaitu yang mengungkapkan seluruh kelas kesalahan.
4. Pengujian yang baik tidak boleh terlalu sederhana atau terlalu kompleks

3.4 Hal Penting dalam Pengujian

Pengujian dilakukan Beberapa hal yang harus diperhatikan dalam melakukan pengujian adalah sebagai berikut :

1. Kualitas dari proses pengujian menentukan kesuksesan pengujian
2. Mencegah perambatan cacat dengan menguji pada seluruh siklus hidup perangkat lunak
3. Menggunakan alat pengujian/ *Testing tools*
4. Harus ada seseorang ahli yang bertanggung jawab untuk memperbaiki proses pengujian

-
5. Pengujian adalah disiplin professional yang membutuhkan orang terlatih dan berkeahlian
 6. Menumbuhkan tim yang bersikap positif

3.5 Arti Pengujian Bagi Penguji

Seorang penguji berburu errors :

1. Suatu penguji yang baik adalah seseorang yang mempunyai probabilitas yang baik untuk mendeteksi error yang belum ditemukan. Pengujian yang sukses adalah pengujian yang mendeteksi error yang belum ditemukan
2. Berfokus pada error yang terlihat/ada

Seorang penguji adalah destructive tetapi kreatif

Pengujian memerlukan imajinasi, ketekunan dan perasaan yang kuat untuk mencari secara sistematis kelemahan dan mendemonstrasikan kegagalan (*failure*)

Seorang Penguji Mengejar errors bukan pembuat program :

- 2 Yang dicari adalah kesalahan dalam produk, bukan orang yang membuat kesalahan
- 3 Developer harus mengerti bahwa penguji bukan melawan mereka tetapi membantu developer.

Cara Mendeteksi Kesalahan :

1. Dengan memeriksa struktur dan desain internal
2. Dengan memeriksa fungsi dari antarmuka pengguna (*user interface*)
3. Dengan memeriksa sasaran design (*design objective*)
4. Dengan memeriksa permintaan user (*user requirement*)
5. Dengan mengeksekusi program

Klasifikasi Kesalahan Program :

1. Kesalahan bahasa (*language error*)
Kesalahan cara penulisan program (*syntax error*) dan/atau kesalahan tata bahasa (*grammatical error*)
2. Kesalahan sewaktu proses (*run-time error*)
Kesalahan kondisi yang belum terpenuhi atau yang akan menyebabkan program hang dan/crash.
3. Kesalahan logika (*logical error*)
Kesalahan mengartikan keinginan analisis. Tidak terjadi kesalahan program secara sintaksis, tetapi akan menghasilkan sesuatu yang tidak diharapkan.

Istilah Kesalahan :

1. *Defect* berasal dari spesifikasi produk, berarti bahwa dalam proses pembuatannya terjadi kesalahan karena pelaksana lapangan tidak memahami hasil pekerjaan para analisis.
2. Variasi berasal dari keinginan *customer/user*, berarti dalam proses perencanaan perangkat lunak, terdapat keinginan customer yang tidak dimasukkan ke dalam dokumen SRS, atau walaupun keinginan customer itu tercantum dalam SRS, namun diabaikan karena kesalahan dalam mengimplementasikan metode pengembangan perangkat lunak.

Pengertian Kesalahan :

1. *Mistake* : suatu aksi manusia yang menyebabkan hasil tidak benar
2. *Faults* : suatu langkah salah, baik proses atau definisi data dalam program komputer.

Perkembangan dari mistake berpotensi menuju failure

3. *Failure* : Suatu hasil yang salah. Hasil adalah manifestasi dari fault (contoh : crash)
4. *Error* : Jumlah dari hasil yang salah.

Failures & Faults

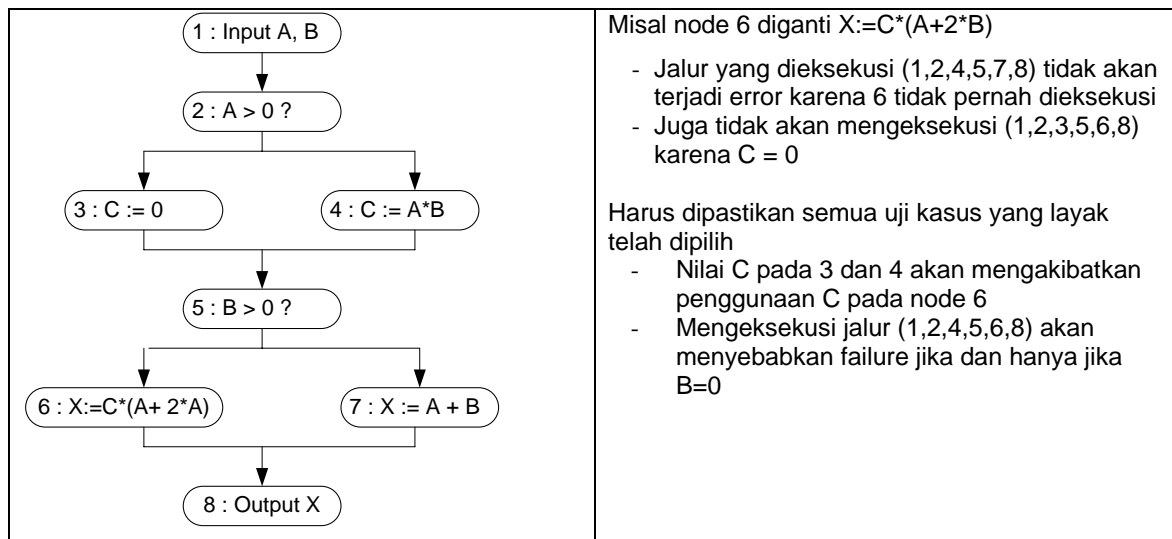
Failure : output yang tidak benar/tidak sesuai ketika sistem dijalankan

Fault : kesalahan dalam source code yang mungkin menimbulkan failure ketika code yang fault tersebut dijalankan.

Tabel 1 – Kelas Kegagalan

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

Contoh Faults, Error & Failures



Gambar 7 – Pendeteksian Faults dan Failure

Kategori Defect :

- 7 *Wrong* → Spesifikasi telah diimplementasikan secara salah (*variances form user*)
- 8 *Missing* → Suatu requirement tertentu tidak dimasukkan ke dalam produk (*Variance from product evaluation*) atau terdapat requirement yang baru ada ketika produk selesai dibuat atau dalam masa pembuatan.

-
- 9 *Extra* → Suatu *requirement* tergabung dalam program tetapi belum / tidak ditentukan (*Variances from specification product*)

Defect vs Failure :

1. Defect adalah hal-hal yang tergabung dalam sistem perangkat lunak
2. *Defect* adalah hal-hal yang tergabung dalam sistem software (dapat ditemukan dalam *software*, dokumentasi dan tata kerja manual), yang pada awalnya tidak mempunyai dampak apapun, hingga akhirnya mempunyai berpengaruh pada user/customer dan pengoperasian sistem (yang disebut cacat)
3. *Defect* yang menyebabkan suatu error dalam pengoperasian atau berdampak negative pada user/customer disebut **Failure**

3.6 Pihak Terkait Pengujian

Pihak Terkait Pengujian :

1. *S/W customer*,
Merupakan kelompok yang mengontrak atau menawarkan jasa agar dibangun suatu s/w
2. *S/W user*
Merupakan kelompok yang akan menggunakan S/w (bagian operasional)
3. *S/W Developer*
Merupakan kelompok yang membangun – merancang dan membuat – s/w
4. *S/W Tester*
Merupakan kelompok yang akan melakukan pengecekan semua fungsi s/w
5. *ISM – Information Services Management*
Merupakan kelompok yang bertanggung jawab memenuhi kebutuhan pelayanan informasi
6. *SOM – Senior Organization Management*
Merupakan kelompok eksekutif untuk pengambilan keputusan
7. *Auditor*
Merupakan kelompok yang mempunyai tanggung jawab untuk mengevaluasi keefektifan, efisiensi dan kecukupan control dalam daerah pelayanan informasi

3.7 Komposisi Tim Penguji

Pendekatan yang dapat dilakukan :

1. Internal Information Services

Anggota	:	Tim Proyek
Kelebihan	:	Meminimalkan biaya Terlatih Memiliki pengetahuan tentang proyek Memudahkan alokasi waktu
Kekurangan	:	Tidak adanya independensi Tidak objektif

2. External Information Services

Anggota	:	Quality assurance Penguji profesional
Kelebihan	:	Pandangan yang independent Merupakan professional di bidangnya Terdapat berbagai orang yang berpengalaman
Kekurangan	:	pembiayaan mahal Terdapat kompetisi antara tim pengembang dan penguji Ketidakpercayaan antara kedua tim

3. Non - Information Services

Anggota	:	User, Auditor, Konsultan
Kelebihan	:	Pandangan yang independent Penilaian yang independent Kemampuan tim untuk mendeteksi kesalahan
Kekurangan	:	Pembiayaan lebih mahal Tidak memiliki pengetahuan tentang informatika Tidak mengerti tentang proyek yang dikerjakan Pandangan yang independent

4. Kombinasi

Anggota	:	Semua pihak yang berkepentingan
Kelebihan	:	Kemampuan yang beraneka ragam Latar belakang pendidikan Pengaruh sekelompok orang
Kekurangan	:	Pembiayaan yang mahal Penjadwalan untuk pertemuan Perbedaan latarbelakang

3.8 Mengkuantifikasi Biaya Menghilangkan Defect

Suatu lembaga Quality Assurance di Amerika Serikat, melaporkan hasil survei-nya, bahwa 60 defect terjadi pada setiap 1000 kode statement, dan 2/3 defect terjadi pada fase requirement hingga desain pada pengembangan sistem.

Penyebab Defect

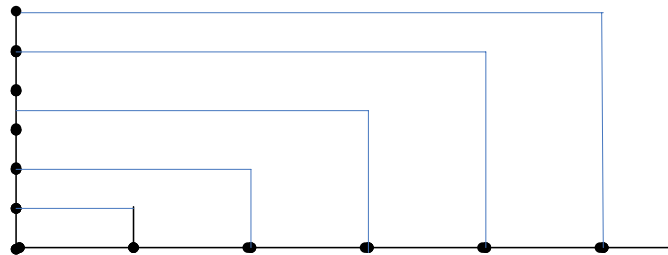
1. Requirement tidak cukup dimengerti
2. User menyatakan requirement secara salah.
3. Requirement dituliskan secara salah
4. Spesifikasi desain tidak tepat
5. Spesifikasi program tidak tepat
6. Kode program salah

7. Instruksi atau struktur program salah
8. Pengujian data salah
9. Pengujian salah dilakukan
10. Pengkoreksian mistake yang salah
11. Pengkoreksian defect menyebabkan defect di bagian lain

Klasifikasi Biaya Defect

1. Defect yang terjadi selama fase requirement hingga desain, maka biaya pengujian dianggap sama dengan biaya pengembangan (faktor 1)
2. Defect yang terjadi selama fase pengujian sistem maka biaya dikali faktor 10
3. Defect yang terjadi setelah sistem berjalan maka biaya dikalikan faktor 100

Contoh Soal :



Gambar 8 – Grafik biaya pengujian

Perhitungan :

	Angka	Faktor	Skala	Total	Akumulasi
Analisis	5 .	1 .	100	= 500	500
Desain	10 .	1 .	100	= 1000	1500
Code	18 .	10 .	100	= 18000	19500
Pengujian Sistem	25 .	10 .	100	= 25000	44500
Instalasi	30 .	10 .	100	= 30000	74500

Biaya Pengujian
(Skala ratusan)

30

25

20

15

10

BAB 4

PELAKSANAAN PENGUJIAN

Strategi pengujian dilakukan untuk mengintegrasikan metode perancangan kasus pengujian software ke dalam langkah-langkah terencana yang tersusun rapi sehingga menghasilkan konstruksi software yang sukses. Yang terpenting adalah strategi pengujian software menyediakan jalan bagi software developer, organisasi penjamin kualitas dan pelanggan karena mendeskripsikan langkah-langkah yang akan dipakai sebagai bagian dari pengujian.

Langkah-langkah ini direncanakan dan kemudian dijalankan sehingga dapat diketahui berapa banyak usaha, waktu dan sumber daya yang akan diperlukan. Oleh karena itu, strategi pengujian manapun harus menyertakan perencanaan pengujian, desain kasus pengujian, pelaksanaan pengujian dan koleksi serta evaluasi data resultan.

Sasaran dari pengujian adalah mengurangi resiko yang terlihat dalam system computer. Strategi pengujian harus mengacu pada resiko dan memberikan proses yang dapat mengurangi resiko tersebut.

4.1 Karakteristik Umum

Karakteristik yang ditemui dalam menentukan strategi pengujian adalah :

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

4.2 Strategi Pengujian

Jenis pengujian yang dapat dilakukan adalah :

1. Pengujian unit program

Pengujian difokuskan pada unit terkecil dari suatu modul program. Dilaksanakan dengan menggunakan driver dan stub. *Driver* adalah suatu program utama yang berfungsi mengirim atau menerima data kasus uji dan mencetak hasil dari modul yang diuji. *Stub* adalah modul yang menggantikan modul sub-ordinat dari modul yang diuji.

2. Pengujian integrasi

Pengujian terhadap unit-unit program yang saling berhubungan (terintegrasi) dengan fokus pada masalah interfacing. Dapat dilaksanakan secara top-down integration atau bottom-up integration.

3. Pengujian validasi

Pengujian ini dimulai jika pada tahap integrasi tidak ditemukan kesalahan. Suatu validasi dikatakan sukses jika perangkat lunak berfungsi pada cara yang diharapkan oleh pemakai.

4. Pengujian sistem

Pengujian yang dilakukan sepenuhnya pada sistem berbasis komputer.

Jenis pengujian yang dilakukan pada saat melakukan pengujian sistem, yaitu :

- *Recovery testing*

Pengujian dilakukan dimana sistem diusahakan untuk gagal, kemudian diuji kenormalannya.

- *Security testing*

Dilakukan untuk menguji mekanisme proteksi

- *Stress testing*

Pengujian yang dirancang untuk menghadapi suatu perangkat lunak kepada situasi yang tidak normal.

- *Performance testing*

Pengujian dilakukan untuk mengetahui kinerja dari sistem

Hal-hal yang diujikan selama pengujian :

1. **Antarmuka modul**

Memastikan bahwa informasi yang berasal dari dan keluar dari modul yang diuji mengalir dengan benar

2. **Struktur Data Lokal**

- 1) Memastikan bahwa data yang disimpan sementara menjaga integritasnya selama eksekusi perintah dalam modul
- 2) Mencari kesalahan-kesalahan dalam bentuk :
 - a) Penggunaan tipe data yang tidak tepat
 - b) Inisialisasi yang salah atau nilai pasti (*default*) yang salah
 - c) Nama peubah yang salah
 - d) *Underflow, overflow, dan addressing exceptions*

3. **Kondisi Batas**

Memastikan bahwa modul beroperasi dengan benar pada batas-batas pemrosesan yang ditentukan

4. **Jalur-jalur Bebas**

- 1) Memastikan bahwa semua kemungkinan jalur kontrol yang mungkin dieksekusi dengan benar paling tidak sekali
- 2) Mencari kesalahan-kesalahan dalam bentuk :
 - a) Penghitungan/pemrosesan yang salah
 - b) Perbandingan yang salah
 - c) Jalur kontrol yang tidak tepat

5. **Jalur-jalur penanganan kesalahan**

- 1) Perancangan perangkat lunak yang baik menuntut agar kondisi salah dapat diantisipasi dan memiliki penanganan kesalahan agar pemrosesan dapat berhenti dengan bersih (*antibugging*)
 - Yourdon
- 2) Mencari kesalahan-kesalahan dalam bentuk :
 - a) Jenis kesalahan tidak dapat dipahami
 - b) Pemberitahuan kesalahan tidak sesuai dengan kesalahan yang dialami

- c) Kondisi kesalahan menyebabkan intervensi sistem sebelum penanganan kesalahan dilakukan
- d) Penanganan kesalahan tidak benar
- e) Jenis kesalahan tidak memberikan informasi yang cukup untuk mencari sumber kesalahan

4.3 Langkah Pengujian

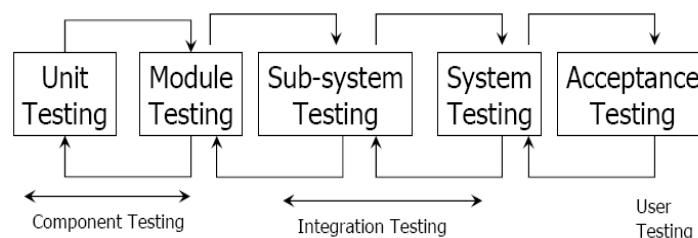
Ketika akan melakukan pengujian terdapat beberapa hal yang harus diperhatikan, yaitu :

1. Menspesifikasikan kebutuhan (*requirement*) produk dalam bentuk yang dapat diukur (*quantifiable*) jauh sebelum pengujian dimulai.
2. Menyatakan tujuan pengujian secara eksplisit.
3. Memahami pengguna perangkat lunak dan membuat profil dari tiap kategori pengguna.
4. Membuat rencana pengujian yang menekankan pada *rapid cycle testing*.
5. Membuat perangkat lunak *robust* yang dapat menguji dirinya sendiri.
6. Menggunakan *formal technical review* sebagai penyaring sebelum pengujian dilakukan.
7. Menggunakan *formal technical review* untuk menilai strategi pengujian dan kasus uji.
8. Mengembangkan ancangan peningkatan berlanjut untuk proses pengujian.

4.4 Proses Pengujian

Tipe-tipe pengujian yang dapat dilakukan (d disesuaikan dengan gambar – 4), yaitu :

1. System Testing
Pengujian terhadap integrasi sub-system, yaitu keterhubungan antar sub-system
2. Acceptance Testing
 - 1) Pengujian terakhir sebelum sistem dipakai oleh user.
 - 2) Melibatkan pengujian dengan data dari pengguna sistem.
 - 3) Biasa dikenal sebagai “alpha test” (“beta test” untuk software komersial, dimana pengujian dilakukan oleh potensial customer)
3. Component testing
 - 1) Pengujian komponen- komponen program
 - 2) Biasanya dilakukan oleh component developer (kecuali untuk system kritis)
4. Integration testing
 - 1) Pengujian kelompok komponen-komponen yang terintegrasi untuk membentuk sub-system ataupun system
 - 2) Dilakukan oleh tim penguji yang independent
 - 3) Pengujian berdasarkan spesifikasi sistem



Gambar – 9 Keterkaitan antar pengujian

BAB 5

TAHAPAN PENGUJIAN

Langkah-langkah yang seharusnya dilakukan ketika melakukan pengujian, adalah :

1. Menentukan apa yang akan diukur melalui pengujian
2. Menentukan bagaimana pengujian akan dilaksanakan
3. Membangun suatu kasus uji (test case), yaitu sekumpulan data atau situasi yang akan digunakan dalam pengujian.
4. Menentukan hasil yang diharapkan atau hasil sebenarnya
5. Menjalankan kasus pengujian
6. Membandingkan hasil pengujian dan hasil yang diharapkan.

Di awal melakukan pengujian, hal pertama yang harus dilakukan adalah membuat rencana pengujian, yang mencakup hal-hal berikut :

- Proses testing
Mendeskripsi fase-fase utama dalam pengujian
- Pelacakan Kebutuhan
Menentukan semua kebutuhan user yang akan diujikan secara individu
- Item yg diuji
Menspesifikasi komponen sistem yang diuji
- Jadwal testing
- Prosedur Pencatatan Hasil dan Prosedur
- Kebutuhan akan Hardware dan Software
- Kendala-kendala
Menentukan hal-hal yang akan mengganggu pengujian, seperti : kekurangan staff, alat, waktu dll.

5.1 Pengujian Tahapan Analisis

Pengujian pada tahapan ini lebih menekankan pada validasi terhadap kebutuhan perangkat lunak, untuk menjamin bahwa kebutuhan telah dispesifikasikan dengan benar. Tujuan pengujian pada tahap ini adalah untuk mendapatkan kebutuhan yang layak dan untuk memastikan apakah kebutuhan tersebut sudah dirumuskan dengan baik.

Faktor-faktor pengujian yang dilakukan meliputi :

- Kebutuhan yang berkaitan dengan metodologi
- Pendefinisian spesifikasi fungsional
- Penentuan spesifikasi kegunaan
- Penentuan kebutuhan portabilitas
- Pendefinisian antar muka sistem.

5.2 Pengujian Tahapan Desain

Pengujian pada tahapan ini bertujuan untuk menguji struktur perangkat lunak yang diturunkan dari kebutuhan perangkat, kebutuhan yang bersifat umum dirinci menjadi bentuk yang lebih spesifik .

Faktor-faktor pengujian yang dilakukan meliputi :

- Perancangan yang berkaitan dengan kebutuhan
- Kesesuaian perancangan dengan metodologi dan teori.
- Portabilitas rancangan
- Perancangan yang dirawat
- Kebenaran rancangan berkaitan dengan fungsi dan aliran data.
- Kelengkapan perancangan antar muka.

5.3 Pengujian Tahapan Implementasi

Merupakan pengujian unit-unit yang dibuat sebelum diintegrasikan menjadi aplikasi secara keseluruhan.

Faktor-faktor pengujian tahap implementasi meliputi :

- Kendali integritas data
- Kebenaran program
- kemudahan pemakaian
- Sifat coupling
- Pengembangan prosedur operasi.

5.4 Pengujian Tahapan Pengujian

Pengujian pada tahapan ini dilakukan untuk menilai apakah spesifikasi program telah ditulis menjadi instruksi-instruksi yang dapat dijalankan pada mesin/komputer. Selain itu, juga untuk menilai apakah instruksi yang ditulis tersebut telah sesuai dengan spesifikasi program.

Faktor-faktor pengujian tahap ini meliputi :

- Pengujian fungsional
- Dukungan manual
- Kemudahan operasi.

BAB 6

FAKTOR PENGUJIAN

Faktor pengujian adalah hal-hal (factor-faktor) yang diperhatikan selama pengujian. Terdapat 15 faktor di dalam pengujian, tetapi tidak semua factor yang mungkin digunakan, hal ini bergantung pada sistem yang diuji.

6.1 Faktor Pengujian

Faktor-faktor pengujian :

1. Reliability

Menekankan bahwa aplikasi akan dilaksanakan dalam fungsi sesuai yang diminta dalam periode waktu tertentu. Pembetulan proses tersangkut kemampuan sistem untuk memvalidasi proses secara benar.

2. Authorization

Menjamin data diproses sesuai dengan ketentuan manajemen. Authorisasi menyangkut proses transaksi secara umum dan khusus.

3. File Integrity

Menekankan pada data yang dimasukkan melalui aplikasi akan tidak bisa diubah. Prosedur yang akan memastikan bahwa file yang digunakan benar dan data dalam file tersebut akan disimpan sekuensial dan benar.

4. Audit Trail

Menekankan pada kemampuan untuk mendukung proses yang terjadi. Pemrosesan data secara keseluruhan berdasarkan retensi dari kejadian yang cukup mendukung keakuratan, kelengkapan, batas waktu dan otorisasi data.

5. Continuity of processing

Menekankan kemampuan untuk meneruskan proses, ketika terjadi suatu permasalahan, dengan menetapkan prosedur yang diperlukan dan back-up informasi untuk melindungi operasi yang mungkin hilang karena masalah tersebut.

6. Service Levels

Menekankan bahwa hasil yang diinginkan didapat dalam waktu yang diinginkan oleh user. Untuk mencapai keinginan tersebut, harus dilakukan penyesuaian antara keinginan user dengan sumber daya yang ada. (Sumber daya mencakup kemampuan input/output, fasilitas komunikasi, pemrosesan dan kemampuan sistem dari software.

7. Access control

Menekankan sumberdaya sistem harus dilindungi dari kemungkinan modifikasi, pengrusakan, penyalahgunaan dan Prosedur keamanan harus dijalankan secara penuh untuk menjamin integritas data dan program aplikasi.

8. Metodology

Menekankan bahwa aplikasi dirancang sesuai dengan strategi organisasi, kebijaksanaan, prosedur, dan standar. Permintaan tersebut, harus diidentifikasi, diimplementasikan dan dipelihara, sesuai dengan permintaan aplikasi.

9. Correctness

Menjamin pada data yang dimasukkan, proses dan output yang dihasilkan dari aplikasi harus akurat dan lengkap. Kelengkapan dan akurasi akan dicapai melalui kontrol transaksi dan elemen data

10. Ease of use

Menekankan perluasan usaha yang diminta untuk belajar, mengoperasikan dan menyiapkan inputan, dan menginterpretasikan output dari sistem. Faktor ini tersangkut dengan usability sistem terhadap interaksi antara manusia dan sistem.

11. Maintainable

Usaha yang diminta untuk mengalokasi dan memperbaiki suatu error dalam pengoperasian sistem. Error dalam hal defect sistem dan salah mengartikan keinginan user.

12. Portable

Usaha yang diminta untuk mengirimkan program dari satu konfigurasi H/W dan atau lingkungan sistem software ke lingkungan yang lain. Termasuk ke dalamnya konversi data, perubahan program, sistem oprasi dan perubahan dokumentasi.

13. Coupling

Usaha yang diminta untuk menghubungkan komponen di dalam sistem aplikasi dan dengan sistem aplikasi yang lain dalam lingkungan pemrosesan.

14. Performance

Jumlah perhitungan sumberdaya dan kode yang diminta sistem untuk melakukan fungsinya, termasuk ke dalamnya kerja manual dan otomatis.

15. Ease of operations

Sejumlah usaha yang diminta untuk mengintegrasikan sistem ke dalam lingkungan operasi dan lingkungan sistem aplikasi, berupa prosedur manual dan otomatisasi.

6.2 Contoh Pengujian

Berdasarkan factor uji :

1. Reliability

- a) Menentukan toleransi.
- b) Desain control dan integritas data
- c) Implementasi control dan integritas data
- d) Pengujian regresi, pengujian manual dan pengujian fungsional
- e) Verifikasi dan ketepatan dan kelengkapan instalasi
- f) Update ketepatan kebutuhan

2. Authorization

- a) Identifikasi aturan otorisasi
- b) Desain aturan otorisasi
- c) Implementasi aturan otorisasi
- d) Pengujian kesesuaian
- e) Mencegah perrubahan data selam instalasi
- f) Menjaga aturan otorisasi

-
3. File Integrity
 - a) Identifikasi kebutuhan integritas file
 - b) Desain control dan integritas file
 - c) Implementasi control dan integritas file
 - d) Pengujian fungsional
 - e) Verifikasi integritas dari produksi file
 - f) Menjaga integritas file
 4. Audit Trail
 - a) Identifikasi kebutuhan rekonstruksi
 - b) Desain audit trail
 - c) Implementasi audit trail
 - d) Pengujian fungsional
 - e) Menyimpan audit trail selama instalasi
 - f) Update audit trail
 5. Continuity of processing
 - a) Identifikasi akibat dari kegagalan
 - b) Desain contingency plan
 - c) Menyusun contingency plan dan prosedurnya
 - d) Pengujian pemulihan
 - e) Memastikan integritas dari pengujian sebelumnya
 - f) Update contingency plan
 6. Service Levels
 - a) Identifikasi tingkat layanan yang diinginkan
 - b) Desain metode untuk mencapai tingkat layanan
 - c) Desain system untuk mencapai tingkat layanan
 - d) Pengujian beban lebih
 - e) Implementasi rencana pencegahan kegagalan instalasi
 - f) Menjaga tingkat layanan
 7. Access control
 - a) Identifikasi hak akses
 - b) Desain Prosedur akses
 - c) Implementasi proseddur keamanan
 - d) Pegujian kesesuaian
 - e) Kontrol akses selama instalasi
 - f) Menjaga keamanan
 8. Metodology
 - a) Penyesuaian kebutuhan dengan methodology
 - b) Penyesuaian desain dengan methodology
 - c) Penyesuaian program dengan methodology
 - d) Penyesuaian pengujian dengan methodology

-
- e) Penyesuaian integrasi dengan methodology
 - f) Penyesuaian perawatan dengan methodology
9. Correctness
- a) Identifikasi spesifikasi fungsional
 - b) Penyesuaian desain dengan requirement
 - c) Penyesuaian program dengan desain
 - d) Pengujian fungsional
 - e) Ketepatan penempatan program dan data pada produksi
 - f) Update kebutuhan
10. Ease of use
- a) Identifikasi spesifikasi kegunaan
 - b) Desain penggunaan fasilitas
 - c) Penyesuaian program dengan desain
 - d) Pengujian dukungan panduan
 - e) Penyebaran kegunaan instruksi
 - f) Menjaga kemudahan penggunaan
11. Maintainable
- a) Identifikasi spesifikasi kegunaan
 - b) Desain dapat dirawat
 - c) Program dapat dirawat
 - d) Inspeksi
 - e) Kelengkapan dokumentasi
 - f) Menjaga keremawatan
12. Portable
- a) Identifikasi kebutuhan protabilitas
 - b) Desain protabilitas
 - c) Penyesuaian program dengan desain
 - d) Disaster testing
 - e) Kelengkapan dokumentasi
 - f) Menjaga protabilitas
13. Coupling
- a) Identifikasi antar muka system
 - b) Kelengkapan desain antarmuka
 - c) Penyesuaian program dengan desain
 - d) Pengujian fungsional dan regresi
 - e) Koordinasi antarmuka
 - f) Memastikan antarmuka yang benar
14. Performance
- a) Identifikasi kriteria performa
 - b) Kriteria pencapaian desain
 - c) Kriteria pencapaian program

-
- d) Pengujian kesesuaian
 - e) Mengawasi performa instalasi
 - f) Menjaga tingkat performa

15. Ease of operations

- a) Identifikasi kebutuhan operasional
- b) Mengkomunikasikan kebutuhan pada operasi
- c) Mengembangkan prosedur operasi
- d) Pengujian operasi
- e) Implementasi prosedur operasi
- f) Update prosedur operasi

6.3 Tingkatan Pengujian

Strategi pengujian perangkat lunak secara berurutan adalah pengujian unit, pengujian terintegrasi dan pengujian system.

6.3.1 Pengujian Unit

Pengujian unit adalah pengujian yang difokuskan pada unit terkecil dari program atau modul. Pengujian ini didasarkan pada informasi dari deskripsi perancangan detil perangkat lunak. Pada umumnya pengujian ini dilakukan secara *white-box* dan *source code based testing* dengan melakukan pengecekan jalur khusus pada struktur kendali modul untuk menyakinkan kelengkapan cakupan dan deteksi maksimum kesalahan.

6.3.2 Pengujian Regresi

Pengujian integrasi adalah pengujian yang difokuskan pada gabungan unit-unit atau modul-modul yang membentuk kesatuan fungsional. Pengujian ini didasarkan pada informasi dari deskripsi prancangan awal perangkat lunak. Pengujian ini dilakukan untuk menemukan kesalahan antarmuka antar modul. Pengujian ini umumnya dilakukan oleh pengembang sendiri atau dilakukan antar pengembang. Pada umumnya, pengujian ini dilakukan secara *white box* dan *black box*.

6.3.3 Pengujian System

Pengujian sistem adalah pengujian yang dilakukan pada sistem komputer secara keseluruhan. Pengujian ini umumnya dilakukan oleh pengembang bersamaan dengan pengembang lain, karena pengujian yang dilakukan berhubungan dengan elemen lain perangkat lunak. Pengujian ini dilakukan untuk mensimulasikan data salah atau data yang berpotensi salah pada antarmuka perangkat lunak.

Pengujian ini dilakukan secara *black box* dan *specification based testing*. Urutan pengujian ini dituangkan dalam perencanaan pengujian yaitu dengan mendefinsikan prosedur pengujian yang kemudian dilanjutkan dengan menentukan data uji.

Pengujian perangkat lunak disebut dengan *alpha testing* dan *beta testing*., yaitu :

1. *Alpha testing* adalah pengujian yang dilakukan oleh pemakai pada lingkungan pengembang, dalam hal ini lingkungan yang terkendali.
2. *Beta testing* adalah pengujian yang dilakukan oleh pemakai pada lingkungan operasi pemakai, dimana lingkungan perangkat lunak tidak lagi dapat dikendalikan oleh pengembang.

BAB 7

METODE PENGUJIAN

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan

Perangkat lunak dapat diuji dengan dua cara, yaitu :

1. Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal, loop, logika, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.
2. Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

7.1 White Box Testing

Pengujian *white box* (*glass box*) adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasi kasus uji tambahan.

Tujuan penggunaan white box untuk menguji semua statement program.

Penggunaan metode pengujian *white box* dilakukan untuk : **{1}** memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali, **(2)** menggunakan semua keputusan logis untuk semua kondisi *true* atau *false*, **(3)** mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi., dan **(4)** menggunakan struktur data internal untuk menjamin validitas jalur keputusan.

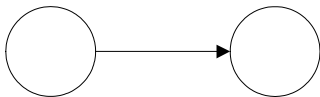
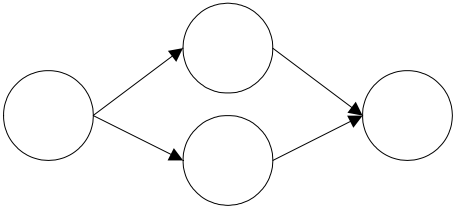
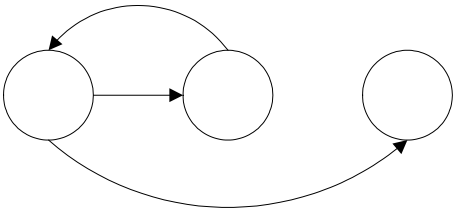
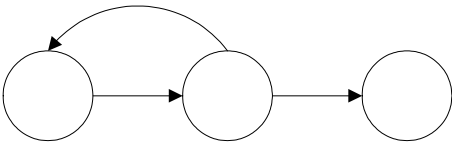
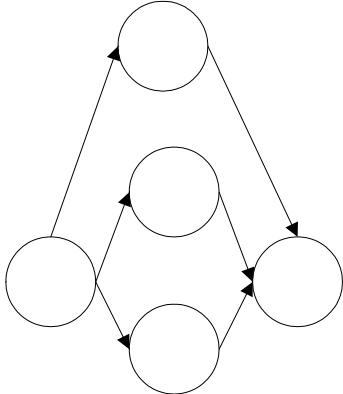
7.1.1 Pengujian Basis Path

Pengujian basis path adalah pengujian white box yang diusulkan pertama kali oleh Tom McCabe. Metode ini memungkinkan penguji dapat mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.

1. Notasi Diagram Alir

Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau node) dan anak panah (link atau edge). Notasi ini menggambarkan aliran control logika yang digunakan dalam suatu bahasa pemrograman.

Tabel 2 – Notasi Diagram Alir

NOTASI	ARTI
	Langkah sequence
	Skema If
	Skema While (..) Do (..)
	Skema Repeat (...) Until (..)
	Skema Case (..)Of

Contoh Suatu PDL

Var

A, B, C : integer

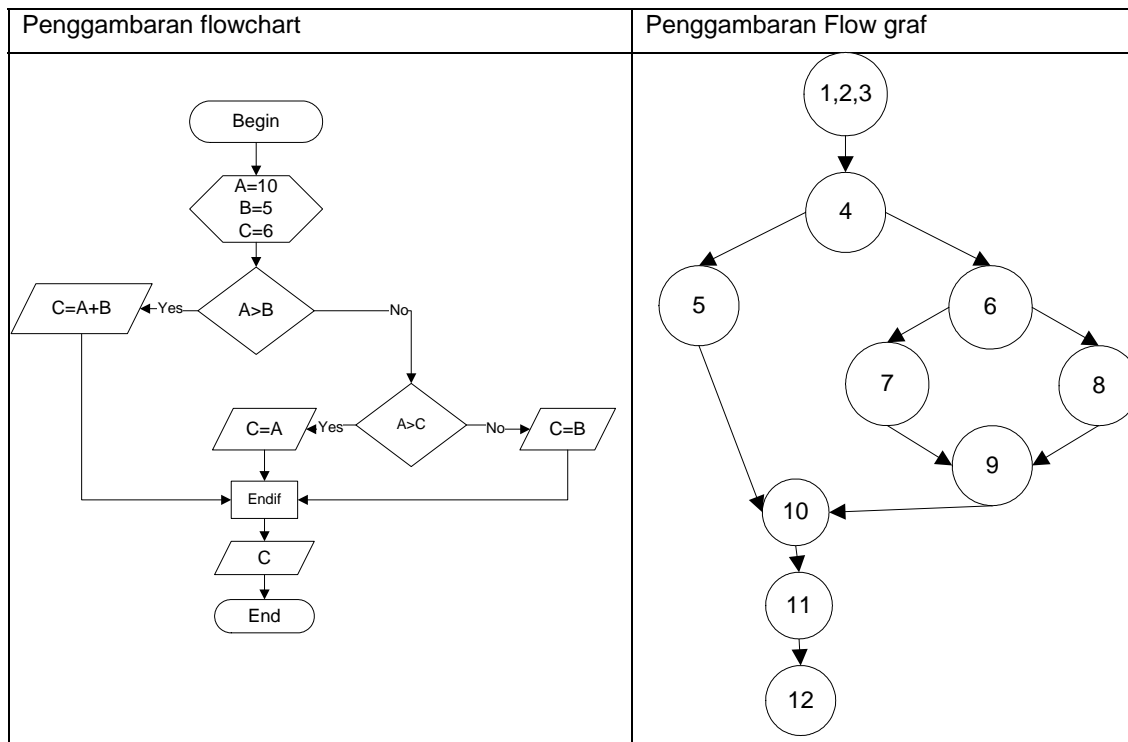
Begin

```

A := 10;           (1)
B := 5;            (2)
C := 6;            (3)
If A > B            (4)
  then C := A + B   (5)
  Else if A > C      (6)
    then C := A     (7)
    Else C := B;    (8)

```

Endif (9)
 Endif (10)
 Writeln('Nilai C = ',C); (11)
 End. (12)



Gambar 10 – Contoh Metode Basis Path

2. Kompleksitas Siklomatis

Kompleksitas Siklomatis adalah metrics perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur yang terdapat dalam program yang mengintroduksi sedikitnya satu rangkaian pernyataan proses atau kondisi baru.

Berdasarkan contoh PDL sebelumnya, maka jalur independent yang didapat :

Jalur 1 : 1,2,3 – 4 – 5 – 10 – 11 – 12

Jalur 2 : 1,2,3 – 4 – 6 – 7 – 9 – 10 – 11 – 12

Jalur 3 : 1,2,3 – 4 – 8 – 9 – 10 – 11 – 12

Penghitungan Siklomatis

1. Region (daerah muka) grafik alir

2. $V(G) = E - N + 2$

E adalah jumlah edge, dan N adalah jumlah node

3. $V(G) = P + 1$

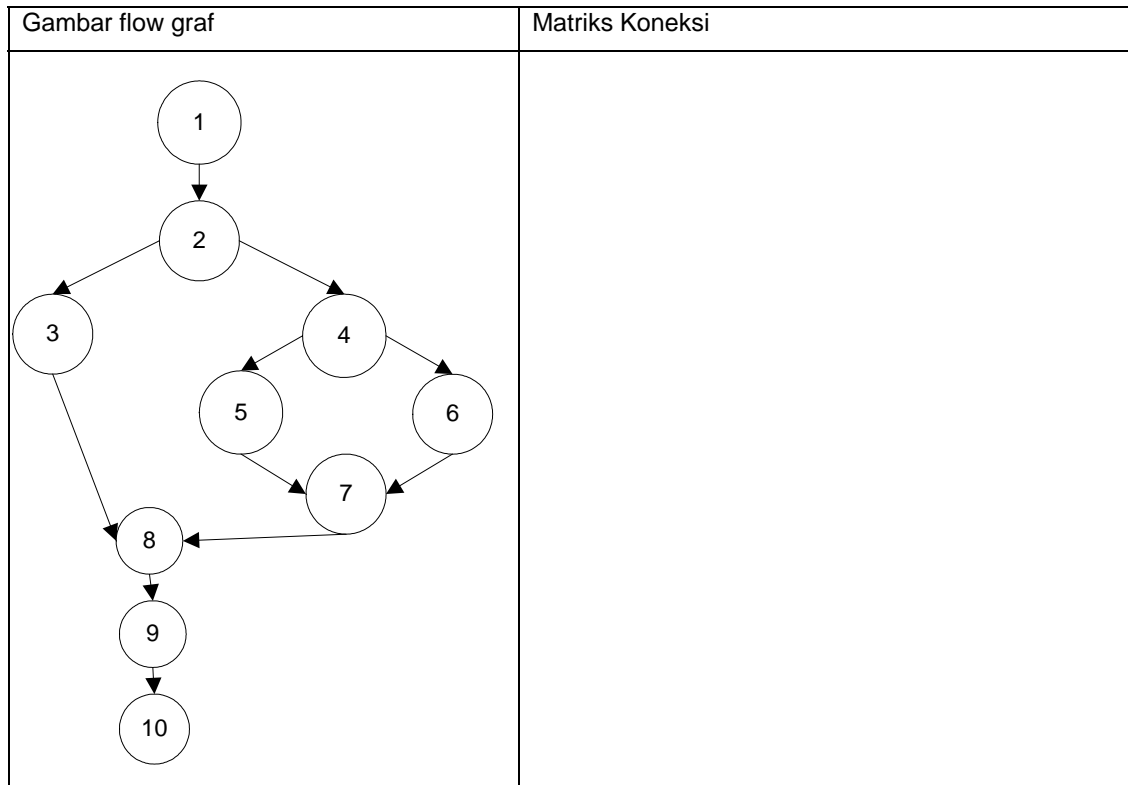
P adalah simpul predikat

Simpul Predikat adalah penggambaran suatu node yang memiliki satu atau lebih inputan, dan lebih dari satu output.

3. Matriks graf

Bentuk struktur data yang sering digunakan untuk menggambarkan pengujian adalah dengan matriks grafis. Matriks grafis adalah matriks bujursangkar yang berukuran sama dengan jumlah simpul pada grafik alir. Inputan dalam matriks harus bersesuaian dengan arah sisi dengan simpul.

Matriks grafis selanjutnya disebut sebagai matriks koneksi, dan digambarkan serupa dengan matriks ketetanggaan dengan memperhatikan arah in-out dari edge.



Gambar 11 – Contoh Matriks Koneksi

7.1.2 Pengujian Struktur Kontrol

Teknik pengujian basis path merupakan salah satu dari sejumlah teknik untuk pengujian struktur control, namun pengujian basis path tidak memadai untuk beberapa kasus uji.

7.1.2.1 Pengujian Kondisi

Pengujian kondisi menggunakan kondisi logis sederhana yang terdapat dalam program. Kondisi sederhana dengan menggunakan variable Boolean, dengan bentuk persamaan.

$$E1 \text{ (operator-relasional) } E2$$

Bila suatu kondisi tidak benar, maka akan terdapat paling tidak satu komponen dari kondisi yang salah, sehingga tipe kesalahan pada suatu kondisi meliputi :

- 1) Kesalahan operator Boolean
- 2) Kesalahan variable Boolean
- 3) Kesalahan tanda kurung Boolean
- 4) Kesalahan operator relasional
- 5) Kesalahan persamaan aritmatika

Contoh :

$$C_1 : B_1 \& B_2$$

B_1 & B_2 adalah dua variable Boolean sehingga akan terdapat 2^2 (2^n) pengujian. Dengan area pengujian :

$B_1 = T ; B_2 = T$	{(T,T), (T,F), (F,T), (F,F)}
$B_1 = T ; B_2 = F$	
$B_1 = F ; B_2 = T$	
$B_1 = F ; B_2 = F$	

7.1.2.2 Pengujian Aliran Data

Metode pengujian aliran data melakukan pengujian dengan menggunakan definisi variable dalam program, efektif digunakan untuk melindungi kesalahan, tetapi akan memiliki cakupan pengukuran dan pemilihan jalur uji yang kompleks.

7.1.2.3 Pengujian Loop

Loop atau skema pengulangan sering digunakan dalam pembuatan program. Terdapat beberapa macam loop, yaitu :

1. Loop Sederhana

Pengujian yang dilakukan harus memperhatikan hal-hal berikut :

- Mengabaikan keseluruhan loop
- Hanya terdapat satu jalur yang melewati loop
- Suatu variable akan melewati loop jika bernilai lebih besar dari nilai yang ditentukan
- Harus memperhatikan batasan variable pada loop. (kondisi : $n-1$, n , $n+1$)

2. Loop Tersarang

Jumlah pengujian akan bertambah secara geometris sesuai jumlah persarangan loop yang ada. Untuk menyederhanakan pengujian, maka harus diperhatikan hal-hal berikut :

- Memulai pengujian dari loop terdalam, dengan memulai pengujian dari nilai minimum
- Melakukan pengujian loop sederhana dari loop terdalam hingga loop terluar, dengan memperhatikan parameter yang digunakan.

3. Loop Terangkai

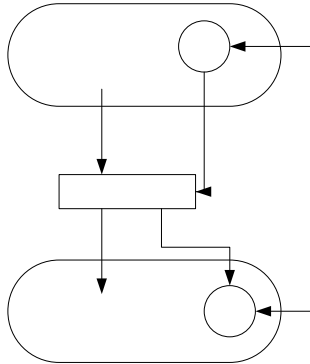
Pengujian untuk loop terangkai harus disesuaikan dengan independensi variable antara loop tersebut. Jika variable yang digunakan dalam loop kedua tidak bergantung dengan loop pertama, maka digunakan pengujian loop sederhana, sedangkan bila loop kedua bergantung secara nilai dengan loop kedua, maka lakukan pengujian tersarang.

4. Loop Tidak Terstruktur

Sama sekali tidak dianjurkan untuk digunakan dalam membuat program.

7.2 Black Box Testing

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian black box diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian black box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.



Gambar 12 – Black Box

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian black box berusaha menemukan kesalahan dalam kategori : **(1)** fungsi-fungsi yang tidak benar atau hilang, **(2)** kesalahan interface, **(3)** kesalahan dalam struktur data atau akses database eksternal, **(4)** kesalahan kinerja, **(5)** inisialisasi dan kesalahan terminasi.

Berbeda dengan pengujian white box, pengujian black box cenderung diaplikasikan selama tahap akhir pengujian. Pengujian black box harus dapat menjawab pertanyaan sebagai berikut :

- Bagaimana validitas fungsional diuji
- Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
- Apakah system akan sangat sensitive terhadap harga input tertentu
- Bagaimana batasan dari suatu data diisolasi
- Kecepatan data apa dan volume data apa yang akan ditoleransi oleh system
- Apa pengaruh kombinasi tertentu dari data terhadap system operasi.

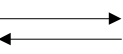
7.2.1 Graf Based Testing

Langkah pertama pada pengujian black box adalah memahami objek yang terdapat dalam model perangkat lunak dan menentukan hubungan yang dimiliki antara objek-objek tersebut. Pengujian berbasis model graf dilakukan terhadap perilaku system.

Graf Based Testing menggambarkan graf yang mewakili hubungan antar objek pada modul sehingga tiap objek dan hubungannya dapat diuji. Pengujian ini dimulai dari mendefinisikan semua

simpul dan bobot simpul, dimana objek dan atribut diidentifikasi, serta memberikan indikasi titik mulai dan berhenti.

Tabel 3 – Notasi Pengujian berbasis graf

NOTASI	ARTI
	Simpul atau node Menggambarkan suatu objek
	Link Menggambarkan hubungan antar objek
	Node weight Menggambarkan properti atau nilai dari data
	Link weight Menggambarkan karakteristik link
	Link paralel Menggambarkan hubungan yang berbeda yang dibangun antar simpul
	Link simetris Menggambarkan hubungan dua arah antara dua objek

Terdapat tiga pola link weight, yaitu :

- *Transitivitas*, yaitu hubungan antara tiga objek atau lebih yang menentukan bagaimana pengaruh hubungan tersebut menyebar pada objek yang ditentukan
- *Simetris*, yaitu hubungan antara dua objek secara dua arah
- *Refleksif*, yaitu hubungan yang mengarah pada node itu sendiri atau *loop null*

Beizer menyebutkan beberapa metode pengujian black box yang menggunakan graf, yaitu :

- 1) Transaction Flow Modeling, metode ini menggunakan node sebagai representasi langkah pada transaksi, dan link sebagai representasi hubungan logika antara langkah-langkah tersebut
- 2) Finite state modeling, metode ini menggunakan node sebagai representasi status dan link sebagai representasi transisi. Statechart atau state transition diagram dapat digunakan untuk membuat graf.
- 3) Data flow modeling, metode ini menggunakan node sebagai representasi objek data dan link sebagai transformasi dari satu objek data ke objek data yang lain.
- 4) Timing modeling, metode ini menggunakan node sebagai representasi objek program dan link sebagai hubungan sekuensial antara objek.

7.2.2 Equivalence Partitioning (Partisi ekuivalensi)

Partisi ekuivalensi adalah metode yang membagi domain input dari suatu program ke dalam kelas data, menentukan kasus pengujian dengan mengungkapkan kelas-kelas kesalahan, sehingga akan mengurangi jumlah keseluruhan kasus pengujian.

Bila suatu *link weight* mempunyai pola *transitivitas*, *simetris*, dan *refleksif* maka akan terdapat kelas ekuivalensi. Kelas ekuivalensi merepresentasikan serangkaian kondisi valid dan invalid untuk

Metode pengujian perbandingan digunakan untuk membangkitkan data uji untuk salah satu perangkat lunak, yang kemudian digunakan sebagai masukan pada pengujian perangkat lunak yang lain.

Comparison Testing digunakan untuk system yang menganut redundancy, kasus uji yang dirancang untuk satu versi perangkat lunak dijadikan masukan pada pengujian versi perangkat lunak lainnya, dan diharapkan hasil kedua versi perangkat lunak harus sama. Jika hasil pengujian kedua perangkat lunak tersebut berbeda maka kedua perangkat lunak itu akan dicek untuk mencari yang salah.

BAB 8

PERANGKAT PENGUJIAN

Perangkat pengujian digunakan untuk membantu menentukan apakah factor-faktor pengujian telah cukup dilakukan pada fase kebutuhan. Rekomendasi yang diberikan untuk penggunaan perangkat pengujian adalah alat Bantu walkthrough dan alat Bantu Mantriks Resiko.

8.1 Walkthrough

Tujuan penggunaan alat Bantu walkthrough menciptakan sebuah situasi dimana suatu tim terdiri dari individu yang terlatih yang dapat menolong tim proyek dalam pengembangan solusi proyek. Walk through menggunakan pengalaman dan keputusan review tim sebagai tambahan atau bantuan dalam proses pembangunan, berorientasi pada bimbingan dalam pemecahan masalah dibandingkan dengan pemenuhan pelaksanaan metodologi.

Lima langkah penggunaan alat bantu Walkthrough :

1. Menyusun aturan-aturan dasar

Aturan dasar bersifat umum, walkthrough paling produktif ketika aturan dasar diterapkan sebelum walkthrough yang sesungguhnya. Aturan dasar harus dimengerti oleh tim proyek maupun tim walkthrough. Aturan dasar terdiri dari :

- 1) Ukuran dan susunan tim walkthrough
- 2) Tanggung jawab tim walkthrough (terbatas pada rekomendasi dan pertanyaan),
- 3) Kewajiban tim proyek menjawab seluruh pertanyaan dan merespon rekomendasi,
- 4) Perkiraan waktu dan lokasi walkthrough,
- 5) Kerahasiaan informasi yang didiskusikan,
- 6) Aspek sistem yang tidak dapat dihadapi dan didiskusikan,
- 7) Siapa yang akan menerima hasil walkthrough dan bagaimana hasil tersebut dapat digunakan.

2. Memilih tim/memberitahu partisipan

Tim walkthrough harus diseleksi berdasarkan tujuan yang akan dicapai. Beberapa pihak yang terlibat (misalnya : user, service information dan senior management) dapat merekomendasikan partisipan tim walkthrough. Hal ini berdasarkan kepentingan proyek. Partisipan yang umum ada di dalam tim, adalah :

- (1) Information Services Project Manager/analisis sistem
- (2) Senior management,
- (3) Operations management,
- (4) User
- (5) management,
- (6) Konsultan yang sesuai. (Konsultan auditor internal, administrator database, konsultan komputer independen)

3. Presentasi proyek

Tim proyek mempresentasikan kebutuhan proyek kepada tim walk through, yang direpresentasikan adalah :

- (1) Pernyataan sasaran dan tujuan proyek,
- (2) Latar belakang informasi, termasuk statistic bisnis yang tepat dalam area aplikasi yang sedang berjalan dan yang diajukan
- (3) Daftar pengecualian yang dibuat oleh tim proyek,
- (4) Mendiskusikan tentang alternative yang ada dan yang akan dipilih
- (5) Kebutuhan walkthrough menggunakan transaksi representative sebagai basis untuk walkthrough

4. Pertanyaan/Rekomendasi

Tujuan pertanyaan/rekomendasi adalah untuk membangkitkan diskusi

5. Laporan akhir (optional)

8.2 Matriks Resiko

Matriks resiko adalah suatu alat bantu yang dirancang untuk menilai kecukupan pengendalian dalam sistem komputer. Kendali disini berarti bahwa semua mekanisme. Metode dan prosedur yang digunakan dalam aplikasi. Diperkirakan bahwa sistem otomatisasi dalam pengendaliannya memerlukan setengah usaha dari pengembangan secara keseluruhan. Oleh karena itu, usaha yang dikeluarkan untuk memastikan kelengkapan kendali merupakan hal yang penting bagi kesuksesan dan kredibilitas sistem aplikasi.

Empat langkah alat bantu matriks resiko :

a) Identifikasi tim resiko

Kunci untuk memperoleh matriks resiko yang benar adalah penetapan tim resiko yang benar, siapa yang akan bertanggung jawab untuk melengkapi matriks. Tujuan dari pelengkapan matriks adalah untuk menentukan kelengkapan kebutuhan dan rencana kendali untuk mengurangi resiko hingga level yang dapat diterima.

Tim resiko harus memiliki minimum keahlian :

- 1) Memiliki pengetahuan mengenai aplikasi user,
- 2) Mengerti konsep resiko,
- 3) Mampu mengidentifikasi kendali,
- 4) Terbiasa dengan resiko aplikasi dan resiko information services,
- 5) Mengerti konsep layanan informasi dan perancangan sistem,
- 6) Mengerti prosedur operasi komputer

Kandidat yang termasuk dalam tim resiko minimal satu dari user dan yang lainnya dari : Internal auditor, konsultan resiko, pemroses data, petugas keamanan, dan manajer operasi komputer

b) Identifikasi resiko

Tujuan pertama dari tim resiko adalah untuk mengidentifikasi resiko berorientasi aplikasi, bukan lingkungan, berkaitan dengan sistem aplikasi. (Resiko yang berkaitan dengan seluruh aplikasi).

Terdapat dua metode untuk mengidentifikasi resiko :

1) Skenario analisis resiko

Tim resiko melakukan brainstorming resiko aplikasi yang potensial dengan menggunakan pengalaman, penilaian dan pengetahuan mengenai wilayah aplikasi.

2) Ceklist resiko

Tim resiko diberi sejumlah resiko yang kerap muncul dalam sebuah aplikasi otomasi. Dari daftar tersebut, tim akan memilih resiko yang mungkin.

c) Menetapkan objektif pengendalian (tahap requirement)

Tujuan pengendalian untuk setiap resiko harus ditetapkan, ketika kendali dapat dinyatakan secara terukur, maka pengendalian tersebut untuk mencapai tujuan memiliki kebutuhan untuk menggunakan kendali keputusan. Kecukupan kendali tidak dapat diuji sampai tingkat kehilangan resiko dapat diterima.

d) Mengidentifikasi kendali pada setiap segmen sistem (Tahap perancangan)

Selama tahap perancangan, tim resiko akan mengidentifikasi kendali pada setiap tahap dari setiap aplikasi untuk setiap resiko yang diidentifikasi, yaitu :

- a) Keaslian : pembuatan dokumen sumber ditambah dengan otorisasi berkaitan dengan transaksi keaslian tersebut
- b) Masukan data : transfer informasi ke media yang dapat dibaca mesin,
- c) Komunikasi : perpindahan data dari satu titik dalam system ke titik yang lain (perpindahan dapat berupa manual atau elektronik),
- d) Pemrosesan : aplikasi logika system ke data,
- e) Penyimpanan : penyimpanan data, baik untuk waktu sementara maupun penyimpanan tetap,
- f) Keluaran : translasi data dari media computer ke median yang dapat dimengerti dan digunakan oleh manusia,
- g) Penggunaan : keputusan bisnis tergantung dari hasil pemrosesan system

Penilaian yang digunakan dalam matriks resiko

- 1. Sangat Lengkap → tim proyek telah melakukan lebih dari yang diharapkan untuk kriteria tersebut
- 2. Evaluasi yang lengkap → tim proyek telah melakukan kerja yang cukup untuk memastikan kendali yang layak untuk kriteria tsb
- 3. Penilaian yang tidak lengkap → tim proyek belum melakukan kerja yang cukup dan harus melakukan kerja lebih dalam daerah kriteria
- 4. Tidak dapat dipagani (Not Applicable – NA) → berkaitan dengan tipe aplikasi atau filosofi perancangan sistem dimana implementasi kriteria tidak dapat dipakai untuk aplikasi tersebut.