Dasar-Dasar Testing

- Obyektifitas Testing
- Misi dari Tim Testing
- Psikologi Testing
- Prinsip – prinsip Testing
- Moto Testing

Obyektifitas (lain) Testing

- Obyektifitas lain testing, antara lain:
  - Meningkatkan kepercayaan bahwa sistem dapat digunakan dengan tingkat resiko yang dapat diterima.
  - Menyediakan informasi yang dapat mencegah terulangnya *error* yang pernah terjadi.
  - Menyediakan informasi yang membantu untuk deteksi *error* secara dini.
  - Mencari *error* dan kelemahan atau keterbatasan sistem.
  - Mencari sejauh apa kemampuan dari sistem.
  - Menyediakan informasi untuk kualitas dari produk *software*.

Misi dari Tim Testing

- Misi dari tim testing tidak hanya untuk melakukan testing, tapi juga untuk membantu meminimalkan resiko kegagalan proyek.
- Tester mencari manifestasi masalah dari produk, masalah yang potensial, dan kehadiran dari masalah.
- Mereka mengeksplorasi, mengevaluasi, melacak, dan melaporkan kualitas produk, sehingga tim lainnya dari proyek dapat membuat keputusan terhadap pengembangan produk.
- Tester tidak melakukan pembenahan atau pembedahan kode, tidak mempermalukan atau melakukan komplain pada suatu individu atau tim, hanya menginformasikan.
- Tester adalah individu yang memberikan hasil pengukuran dari kualitas produk.

Psikologi Testing

- Bila pengembangan dilakukan secara konstruktif, maka testing adalah destruktif.
- Seorang pengembang bertugas membangun, sedangkan seorang tester justru berusaha untuk menghancurkan.
- Bila seorang disainer harus menanamkan pada benaknya dalam-dalam akan testabilitas, programer harus berorientasi pada "*zero defect minded*", maka tester harus mempunyai keinginan yang mendasar untuk "membuktikan kode gagal (*fail*), dan akan melakukan apa saja untuk membuatnya gagal."
- Jadi bila seorang tester hanya ingin membuktikan bahwa kode beraksi sesuai dengan fungsi bisnisnya, maka tester tersebut telah gagal dalam
- menjalankan tugasnya sebagai tester.

6 Kunci Prinsip-Prinsip Testing

A. Testing yang komplit tidak mungkin.
B. Testing merupakan pekerjaan yang kreatif dan sulit.
C. Alasan yang penting diadakannya testing adalah untuk mencegah terjadinya *errors*.
D. Testing berbasis pada resiko.
E. Testing harus direncanakan.
F. Testing membutuhkan independensi.

A. Testing yang komplit tidak mungkin

Testing yang komplit tidak mungkin karena memperhatikan pertimbangan-pertimbangan akan hal-hal sebagai berikut:

- **Domain Masukan**

  Melakukan semua tes pada berbagai varian masukan yang ada (valid, tidak valid, yang diedit, variasi berdasarkan waktu)

- **Kompleksitas**

  User interface dan disain sangat komplek untuk dilakukan testing secara komplit .

- **Jalur Program**

  Jalur yang banyak yang mungkin dilewati pada suatu program untuk dites secara komplit.

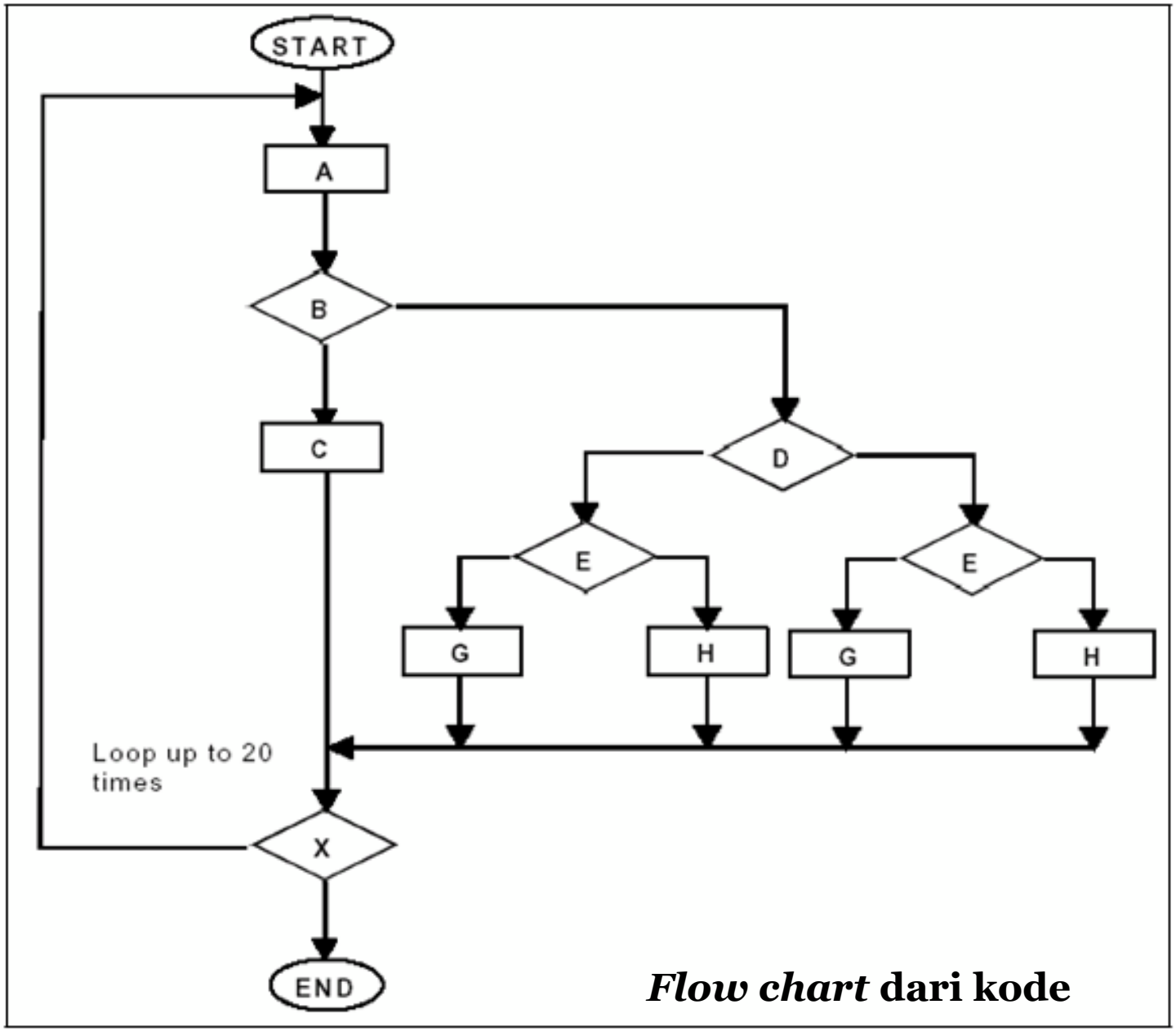
Contoh Jalur Program

- Misal akan dilakukan testing terhadap kode program sebagaimana terdapat pada gambar. Plotkan semua jalur dari awal (START) sampai akhir (END). X akan dapat pergi ke END atau melakukan *loop* kembali ke A 19 kali.
- Ada lima Jalur dari A ke X, yaitu: ABCX, ABDEGX, ABDEHX, ABDFIX, dan ABDFJX. Maka keseluruhan kombinasi jalur yang harus dites adalah 5 + 52 + 53 + … + 520 = 1014 (100 Triliun).

```
Count = 0
Do
        Count = Count + 1                                   A
        Read Record
        If Record <> EOF Then                               B
                If ShortRecord(Record) Then                 D
                        If EarlySortRecord(Record) Then     E
                                ProcessEarlyShort(Record)   G
                        Else
                                ProcessLateShort(Record)    H
                        End If
                Else
                        If EarlyLongRecord(Record) Then     F
                                ProcessEarlyLong(Record)    I

                        Else
                                ProcessLateLong(Record)     J
                        End If
                End If
        Else
                Msgbox "EOF Reached"                        C
        End If
Until Record = EOF Or Count > 20                            X
```

Penggalan kode suatu program



***Flow chart* dari kode**

- Perencanaan tes didominasi dengan kebutuhan untuk memilih sejumlah kecil *test case* dari seluruh kemungkin yang amat sangat banyak.
- Testing tidak untuk membuktikan kebenaran program / sistem, tetapi hanya membuktikan keberadaan *error* dengan kondisi-kondisi yang mempunyai kesamaan secara mendasar dengan yang diteskan.
- Jumlah *error* pada sistem dapat diprediksi dengan akurasi tertentu, tapi tidak dapat menjamin akan tidak adanya lagi *error* lain pada produk selain yang telah diprediksikan.

- **Kesimpulan:**
  - Jadi tidak mungkin melakukan testing pada tiap kemungkinan kombinasi perhitungan secara menyeluruh.
  - Yang mungkin adalah melakukan testing logika dari program dan yakinkan semua kondisi dari semua level komponen telah diperiksa.

B.Testing merupakan pekerjaan yang kreatif dan sulit

- Terdapat mitos yang salah tentang Testing, dimana:
  - Testing itu mudah.
  - Tiap orang akan dapat melakukan testing dengan sendirinya.
  - Tidak dibutuhkan pelatihan atau pengalaman.

- Padahal sebenarnya testing bukanlah suatu hal yang sederhana, karena:
  - Untuk melakukan testing secara efektif, harus mengetahui keseluruhan sistem.
  - Sistem tidak sederhana atau tidak mudah untuk dipahami.

- Untuk dapat sukses dalam melakukan testing dibutuhkan hal-hal penting sebagai berikut:
  - Kreatifitas.
  - Pengetahuan bisnis.
  - Pengalaman testing.
  - Metodologi Testing.

C. Alasan yang penting diadakannya testing adalah untuk mencegah terjadinya *error*.

- Konsep siklus dari testing:
  - Testing bukan untuk satu fase pengembangan saja.
  - Hasil Testing diasosiasikan pada tiap fase pengembangan.
- Semua testing harus dapat dapat dilacak dan memenuhi kebutuhan dari konsumen.
- Obyektivitas dari testing adalah memperbaiki *error*.

D. Testing berbasis pada resiko

- Testing merupakan hasil pertimbangan dari resiko dan ekonomi, dimana secara praktis testing merupakan hasil pertimbangan tarik-ulur dari empat faktor utama:
  - **Sumber daya dan biaya yang dibutuhkan**
  - **Biaya dari keterlambatan pengiriman produk**
  - **Kemungkinan adanya suatu *defect*** (Kurangnya sesuatu yang diperlukan atau diinginkan untuk menyelesaikan sebuah kesempurnaan)
  - **Biaya yang disebabkan oleh *defect***

E.Testing harus direncanakan

- Testing yang baik butuh pemikiran dengan pendekatan secara keseluruhan, disain tes dan penetapan hasil yang diinginkan untuk tiap kasus tes (*test case*) yang dipilih.

- Suatu dokumen yang mencakup keseluruhan dari tujuan testing dan pendekatan testing disebut Rencana Tes (*Test Plan*).

- Suatu dokumen atau pernyataan yang mendefinisikan apa yang telah dipilih untuk dites dan menjelaskan hasil yang diharapkan disebut Disain Tes (*Test Design*).

Contoh:

Rencana Tes	Disain Tes
Pernyataan obyektifitas testing	Spesifikasi tes yang dikembangkan
Deskripsi pendekatan tes	Deskripsi pengelompokan tes
Sekelompok tugas untuk mencapai obyektifitas testing	

Rencana tes dibuat setelah model kebutuhan selesai dibuat. Dan detil dari definisi *test case* dibuat setelah disain model disetujui. Atau dengan kata lain tes direncanakan dan di disain sebelum kode dibuat.

Pentingnya Perencanaan

- Apa yang menjadi penilaian suatu tes tertentu benar?
  - Kepercayaan akan apa yang dapat mereka hasilkan lawan biaya yang mereka pergunakan untuk testing.
  - Adanya penemuan masalah dan *defect*.

- Perencanaan tes sangat penting, karena:
  - Untuk dapat menjaga arah pelaksanaan tes agar tidak menyimpang dari tujuan tes itu sendiri, yaitu untuk mengukur kualitas *software*.
  - Untuk menjaga kesesuaian penggunaan sumber daya dan jadual proyek, dengan menetapkan apa yang akan dites dan kapan berhenti.
  - Untuk membuat *test case* yang baik (tepat guna), dengan menetapkan apa hasil yang diharapkan sehingga akan membantu tester untuk fokus terhadap apa yang akan dites.

F. Testing butuh kebebasan

- Bila menginginkan adanya pengukuran yang tak biasa maka dibutuhkan pula tester yang tak biasa.
- Apa yang disebut Tester yang independen (tak tergantung/bebas):
  - Pengamat yang tidak biasa
  - Orang yang bertujuan untuk mengukur kualitas *software* secara akurat
- Testing yang paling efektif harus dilakukan oleh pihak ketiga.

Isu-isu seputar Testing

- Sistem itu " *Buggy* "
- Testing digambarkan dengan gambaran yang menakutkan
- Batas waktu menjadi hambatan bagi testing
- Testing bukan organisasi dan ilmu
- Manajemen pendukung untuk tes t ing kurang dari ideal
- Testing tidak ditampilkan sebagai suatu karir yang menjanjikan
- Teknologi baru ataupun lama menyulitkan situasi

Testabilitas (1)

- Testabilitas yang harus diperhatikan dalam merekayasa software, yaitu:
  - **Operability** (kemampuan software ditest dengan lebih efisien)
  - **Observability** (kemampuan software yang dapat diamati kode sumbernya jika terdapat kesalahan)
  - **Controllability** (kemampuan software yang dapat dikontrol cth; input,output, variabel, hardware, software)
  - **Decomposability** (kemampuan software yang dibangun dari modul-modul yang independen)

Testabilitas (2)

- Testabilitas yang harus diperhatikan dalam merekayasa software, yaitu:
  - **Simplicity** (kemampuan software dalam kesederhanaan fungsi, kode & struktur)
  - **Stability** (kemampuan software yang tidak mudah mengalami perubahan)
  - **Understandability** (kemampuan software yang mudah untuk dipahami, cth: dokumentasi, komponen & disain)

Kemampuan Tester yang Diharapkan

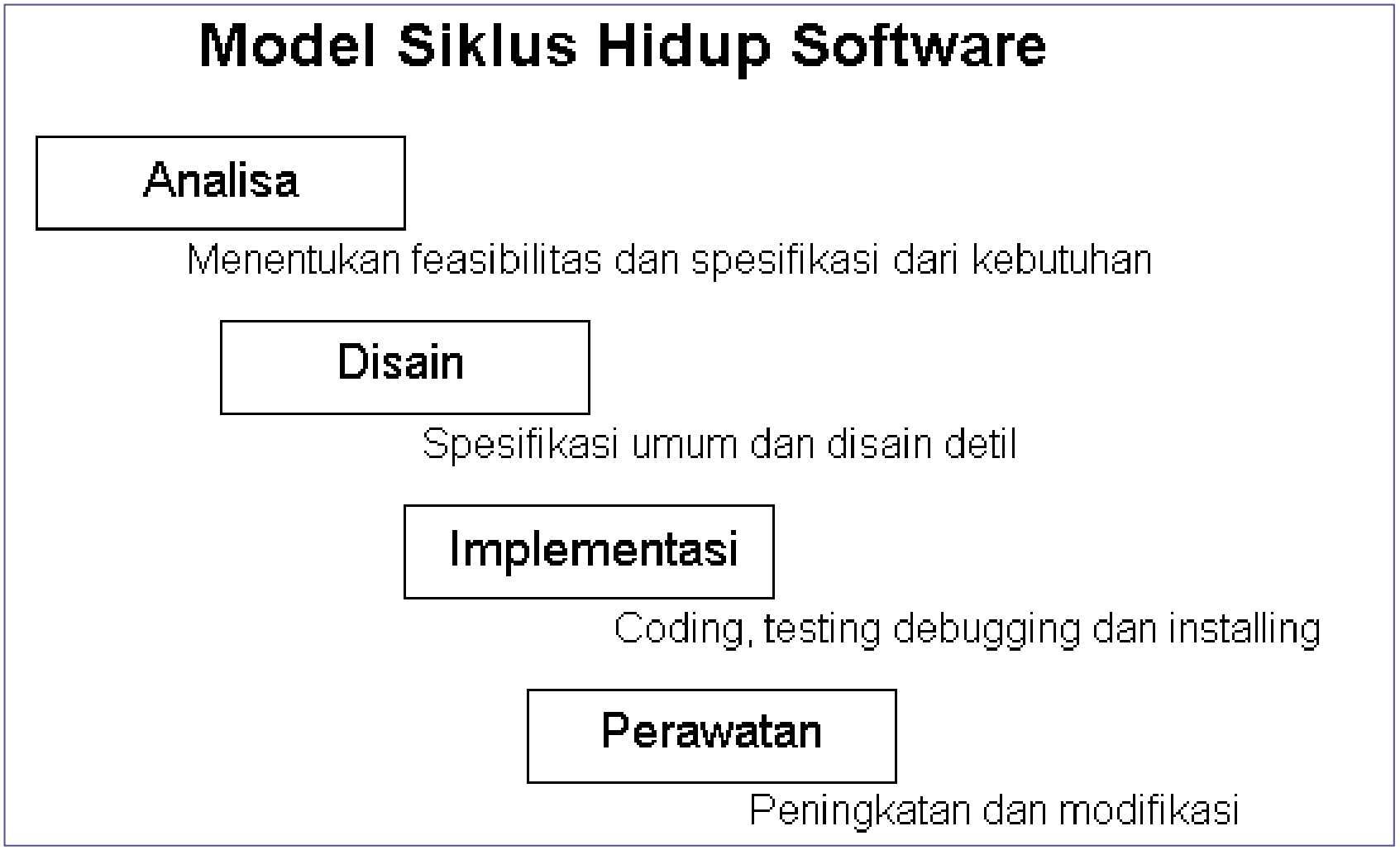
- **Kemampuan secara umum**
  analisa yang kuat dan terfokus,komunikasi yang baik & mempunyai latar belakang QA
- **Pemahaman terhadap metodologi**
  (Pengembangan rencana tes, pembuatan dan perawatan lingkungan tes,standar tes & dokumentasi tes (seperti test cases dan procedure test)
- **Pengetahuan akan pendekatan testing**
  (Integration testing, Acceptance Testing, Stress / Volume Testing, Regression testing, Functional testing, End-To-End Testing, GUI Testing)
- **Pengetahuan tentang sistem**
  (Perbankan/Keuangan, Produk Komersial, Telecom, Internet, Y2K)
- **Pengetahuan dan pengalaman akan penggunaan alat bantu testing**
  (Alat bantu *capture* atau *playback* (seperti WinRunner), Alat bantu *Load testing* (seperti LoadRunner, RoboTest))
- **Kemampuan terhadap linkungan testing**
  (Mainframe (seperti MVS, JCL). Client – Server (seperti WinNT, UNIX))
- **Kemampuan terhadap aplikasi**
  (Dokumentasi (seperti office, excel, word, Lorus Notes), Database (seperti oracle, access, 3GL, 4GL, SQL, RDBMS) Pemrograman (seperti C++, VB, OO))

Pengertian *Defect* dari *Software*

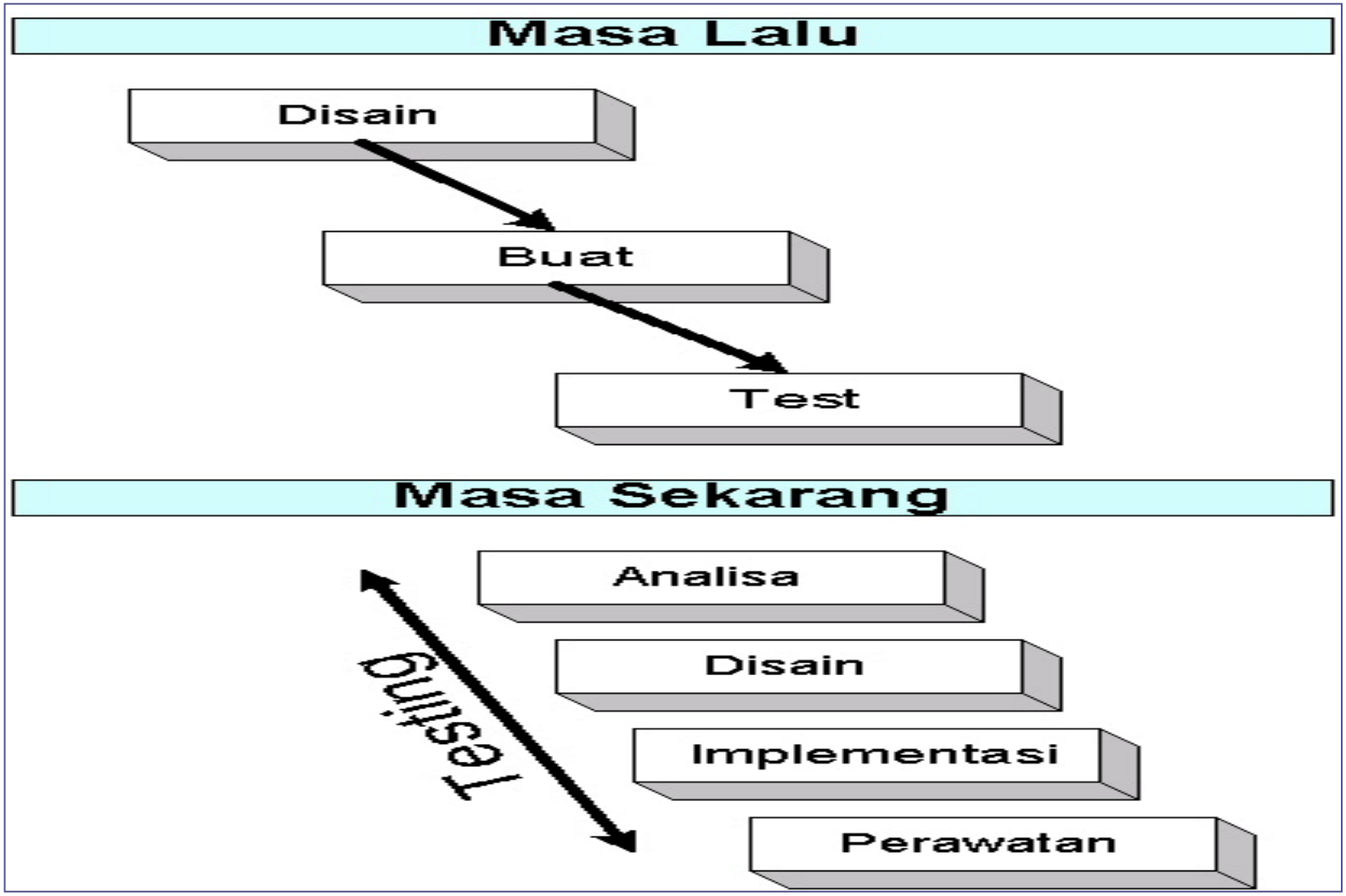
Ada 13 kategori utama *defect* dari *software*, yaitu

- ***User interface errors*** - sistem memberikan suatu tampilan yang berbeda dari spesifikasi.
- ***Error handling*** – pengenalan dan perlakuan terhadap *error* bila terjadi.
- ***Boundary – related errors*** - perlakuan terhadap nilai batasan dari jangkauan mereka yang mungkin tidak benar.
- ***Calculation errors*** - perhitungan arimatika dan logika yang mungkin tidak benar. *Initial and later states* - fungsi gagal pada saat pertama digunakan atau sesudah itu. *Control flow errors* - pilihan terhadap apa yang akan dilakukan berikutnya tidak sesuai untuk status saat ini.
- ***Errors in handling or interpreting data*** - melewatkan dan mengkonversi data antar sistem (dan mungkin komponen yang terpisah dari sistem) dapat menimbulkan *error*.
- ***Race conditions*** - bila dua *event* diproses akan maka salah satu akan diterima berdasarkan prioritas sampai pekerjaan selesai dengan baik, baru pekerjaan berikutnya. Bagaimanapun juga kadang-kadang event lain akan diproses terlebih dahulu dan dapat menghasilkan sesuatu yang tidak diharapkan atau tidak benar.
- ***Load conditions*** - saat sistem dipaksa pada batas maksimum, masalah akan mulai muncul, seperti *arrays, overflow, diskfull*.
- ***Hardware*** - antar muka dengan suatu *device* mungkin tidak dapat beroperasi dengan benar pada suatu kondisi tertentu seperti *device unavailable*.
- ***Source and Version Control*** - program yang telah kadaluwarsa mungkin akan dapat digunakan lagi bila ada revisi untuk memperbaikinya.
- ***Documentation*** - pengguna tak dapat melihat operasi yang telah dideskripsikan dalam dokumen panduan.
- *Testing errors* - tester membuat kesalahan selama testing dan berpikir bahwa sistem berkelakuan tak benar.

Siklus hidup Software secara umum



Siklus hidup Testing



Aktifitas Testing secara Umum

1. Perencanaan
   - Rencana pendekatan umum
   - Menentukan obyektivitas testing
   - Memperjelas rencana umum
2. Akuisisi
   - Disain tes
   - Menerapkan tes
3. Pengukuran
   - Eksekusi tes
   - Cek terminasi
   - Evaluasi hasil

Tiga Tingkatan Testing secara umum

1. ***Unit testing***

   Testing penulisan kode-kode program dalam satuan unit terkecil secara individual.

2. ***System Testing***

   Proses testing pada sistem terintegrasi untuk melakukan verifikasi bahwa sistem telah sesuai spesifikasi.

3. ***Acceptance Testing***

   Testing formal yang dilakukan untuk menentukan apakah sistem telah memenuhi kriteria penerimaan dan memberdayakan pelanggan untuk menentukan apakah sistem dapat diterima atau tidak.

1.Unit Testing

Poin	Keterangan
Tujuan	Konfirmasi bahwa modul telah dikode dengan benar
Pelaku	Biasanya programer
Apa yang di tes	• Fungsi (*Black Box*). • Kode (White Box). • Kondisi ekstrim dan batasan-batasan.
Kapan selesai	Biasanya saat programer telah merasa puas dan tidak diketahui lagi kesalahan
Alat bantu	Tidak biasa digunakan
Data	Biasanya tidak didata

2.System Testing

Poin	Keterangan
Tujuan	Merakit modul menjadi suatu sistem yang bekerja. Dan menentukan kesiapan untuk melakukan *Acceptance Test*.
Pelaku	Pemimpin tim atau grup tes
Apa yang di tes	• Kebutuhan dan fungsi sistem. • Antarmuka sistem.
Kapan selesai	Biasanya bila mayoritas kebutuhan telah sesuai dan tidak ada kesalahan mayor yang ditemukan.
Alat bantu	• Sistem pustaka dan pustaka *test case*. • Generator, komparator dan simulator data testing.
Data	• Data kesalahan yang ditemukan. • Test case.

2. Acceptance Testing

Poin	Keterangan
Tujuan	Mengevaluasi kesiapan untuk digunakan
Pelaku	Pengguna akhir atau agen
Apa yang di tes	• Fungsi mayor. • Dokumentasi • Prosedur.
Kapan selesai	Biasanya bila pengguna telah merasa puas atau tes berjalan dengan lancar / sukses.
Alat bantu	Komparator.
Data	Formalitas dokumen.