BAB I

BUG AND DEBUGGING

1.1 Apakah BUG itu?

Sebuah BUG perangkat lunak dapat didefinisikan sebagai:

Suatu kesalahan pengkodean tidak terduga yang dapat menyebabkan kecacatan, kesalahan, atau ketidaksempurnaan dalam suatu program komputer.

Dengan kata lain, jika suatu program tidak berfungsi sebagaimana mestinya, kemungkinan besar terdapat BUG di dalamnya.

1.2 Mengapa BUG bisa terjadi?

BUG dapat disebabkan oleh beberapa hal berikut:

1. **Pendefinisian kebutuhan perangkat lunak yang tidak jelas.**
2. Selalu adanya **perubahan kebutuhan perangkat lunak.**
3. Juga saat **menyelesaikan BUG** di salah satu bagian / komponen.
4. Disain / disain ulang, antarmuka pengguna, pengintegrasian modul, serta pengelolaan basis data dapat **menambah kompleksitas perangkat lunak** dan sistem secara keseluruhan.
5. Permasalahan mendasar dari **disain perangkat lunak dan arsitekturnya**.
6. **Penjadwalan ulang sumberdaya**, melakukan kembali atau membuang pekerjaan yang sudah selesai, perubahan persyaratan perangkat keras / perangkat lunak juga dapat mempengaruhi perangkat lunak. Menugaskan pengembang baru ke proyek di tengah jalan dapat pula menyebabkan bug.
7. Programer yang selalu **berhadapan dengan tengat waktu** mendekati akhir.
8. **Kompleksitas dalam melacak** BUG itu sendiri dapat menyebabkan timbulnya BUG.

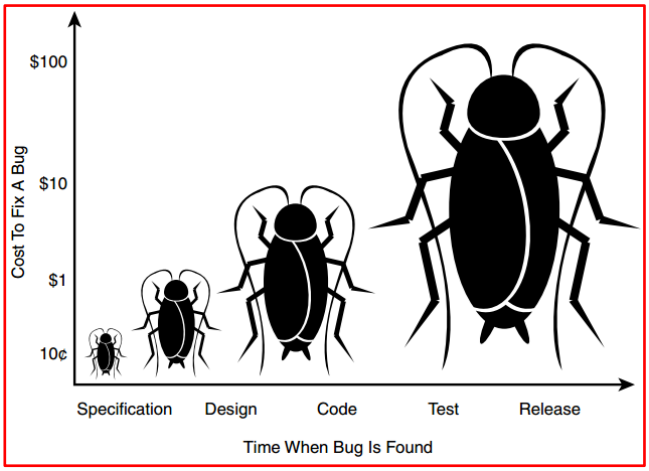
1.3 Siklus hidup BUG

Beberapa fase berbeda dari suklus hidup BUG sebagai berikut:

1. **Open:** Suatu bug berada dalam status “Open” saat penguji telah mengidentifikasi area permasalahan.
2. **Accepted:** Bug dalam status “Accepted” saat pengembang menandainya, lalu pengembang menyetujui validasinya.
3. **Not Accepted / Won’t fix:** Jika pengembang mempertimbangkan BUG sebagai level rendah atau tidak diterima sebagai BUG.
4. **Pending:** Suatu BUG yang disetujui oleh pengembang tetapi tidak langsung diselesaikan segera.

5. **Fixed:** BUG yang telah dikoreksi oleh programmer lalu diberikan status sebagai Fixed.
6. **Close**: BUG yang telah dikoreksi akan ditandai oleh penguji dengan status Close.
7. **Re-Open**: BUG yang telah dikoreksi dapat terbuka kembali oleh penguji saat penyelesaian yang menimbulkan masalah di lain tempat.

1.4 Biaya mengatasi BUG



1.5 When can testing be stopped / reduced?

Susah untuk menentukan kapan waktu yang tepat untuk menghentikan pengujian. Terdapat beberapa faktor yang membantu menentukan kapankah pengujian dapat dihentikan atau dikurangi, yaitu:

1. Tenggat waktu telah tercapai (tenggat waktu rilis, tenggat waktu pengujian, dll.)
2. Kasus uji telah lengkap dengan melewati persentase tertentu
3. Anggaran pengujian telah habis
4. Cakupan kode / fungsionalitas / persyaratan telah mencapai poin yang ditentukan
5. Tingkat / jumlah bug telah sampai di bawah level tertentu.
6. Masa pengujian Beta atau Alpha telah berakhir.

1.6 Istilah-istilah dalam pengujian

- **Bug**: Suatu kesalahan pengkodean tidak terduga yang dapat menyebabkan kecacatan, kesalahan, atau ketidaksempurnaan dalam suatu program komputer. Dengan kata lain, jika suatu program tidak berfungsi sebagaimana mestinya, kemungkinan besar terdapat BUG di dalamnya.
- **Error**: Suatu ketidakcocokan antara program dan spesifikasinya merupakan error dalam program.

- **Defect** (cacat): Merupakan suatu varian dari atribut produk yang diinginkan (dapat berupa data yang salah, hilang, dsb), yang bisa berupa dua hal yaitu defect yang berasal dari produk atau varian dari ekspektasi pengguna. Dalam hal ini, defect dalam sistem perangkat lunak dan tidak berdampak sampai mempengaruhi pengguna dan sistem operasional. Sebanyak 90% dari semua cacat dapat disebabkan oleh masalah proses.
- **Failure**: Suatu defect yang menyebabkan kesalahan dalam pengoperasian atau berdampak negatif pada pengguna / pelanggan.
- **Quality Assurance**: Berorientasi pada pencegahan kecacatan, memastikan semua pihak yang berkepentingan dengan proyek mematuhi proses dan prosedur, standar dan template, serta tinjauan kesiapan dari pengujian.
- **Quality Control**: Kontrol kualitas atau rekayasa kualitas adalah serangkaian tindakan yang diambil untuk memastikan bahwa produk atau layanan yang rusak tidak diproduksi, dan bahwa desainnya telah memenuhi persyaratan performanya.
- **Verification**: Verifikasi memastikan bahwa produk didesain untuk menyajikan semua fungsionalitas kepada pelanggan;
- **Validation**: Validasi memastikan bahwa fungsionalitas (sebagaimana didefinisikan dalam persyaratan) merupakan perilaku dari produk yang dimaksudkan; validasi biasanya melibatkan pengujian secara aktual dan terjadi setelah verifikasi selesai dilakukan.

1.7 Debugging

- *Debugging* terjadi sebagai akibat dari pengujian yang berhasil.
- *Debugging* bukan merupakan pengujian.
- Jika *test case* mengungkap kesalahan, maka *debugging* adalah proses yang menghasilkan penghilangan kesalahan.
- Ada yang mengatakan lain, bahwa *debugging* merupakan proses mental yang dipahami secara buruk yang menghubungkan sebuah *simptom* dengan suatu penyebab.

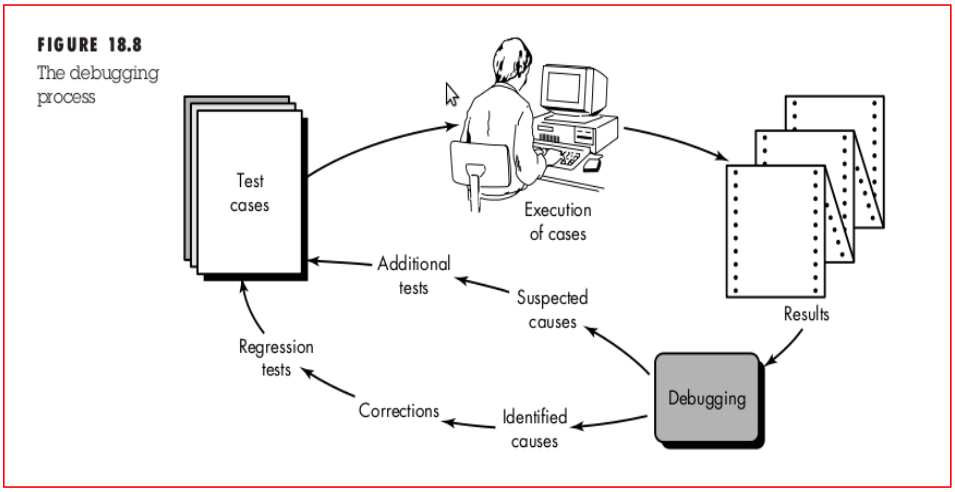
1.7.1 Mengapa Debugging Sulit Dilakukan?

Beberapa karakteristik yang menyebabkan sulitnya debugging dilakukan di antaranya:

- Gejala dan penyebab dapat **jauh secara geografis**
- Gejala dapat **kadang-kadang hilang** ketika kesalahan lain dibetulkan
- Gejala dapat **disebabkan oleh sesuatu yang tidak salah** (misalnya pembulatan yang tidak akurat)
- Gejala dapat **disebabkan oleh kesalahan manusia** yang tidak mudah ditelusuri

- ❖ Gejala dapat merupakan hasil dari **masalah timing**, bukan dari masalah pemrosesan
- ❖ Mungkin **sulit untuk mereproduksi kondisi input** secara akurat (misal: aplikasi real time dimana pengurutan input tidak ditemukan)
- ❖ Gejala dapat **muncul sebentar-sebentar**
- ❖ Gejala dapat **berhubungan dengan penyebab** yang didistribusikan melewati sejumlah tugas yang bekerja pada prosesor yang berbeda.

1.7.2 Proses Debugging



1.7.3 Pertimbangan Psikologis

Menanggapi aspek manusia dari *debugging*, Shneiderman menyatakan:

- ❖ Debugging merupakan salah satu dari bagian pemrograman yang membuat lebih frustasi.
- ❖ Debugging memiliki elemen pemecahan masalah atau pengganggu otak, yang bersama dengan penghindaran kesadaran bahwa Anda melakukan suatu kesalahan.
- ❖ Terdapat kekhawatiran yang meningkat dan keengganan untuk menerima kesalahan akan meningkatkan kesulitan tugas.
- ❖ Ada keluhan yang sangat mendalam mengenai pembebasan dan pengurangan ketegangan hingga pada akhirnya bug... dikoreksi.

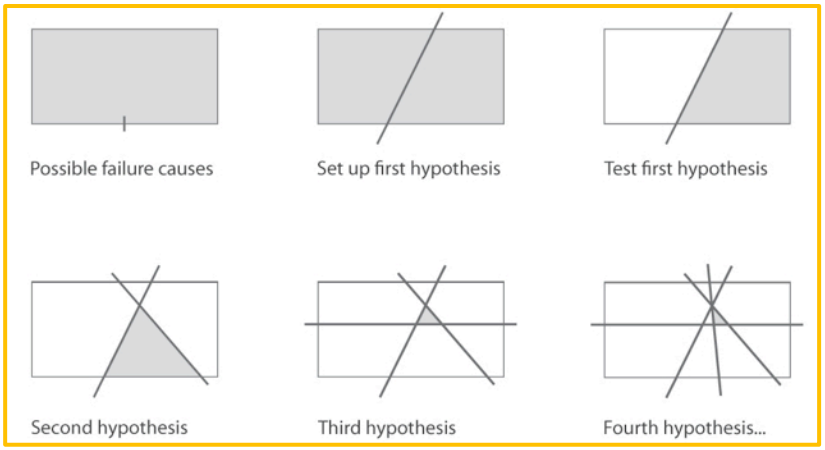
1.8 Tiga Kategori Pendekatan Debugging

Berikut beberapa hal umum yang dilakukan sebagai pendekatan melakukan debugging:

1. **Gaya kasar** (*brute force*)

   Menggunakan filosofi: "Biarkan komputer menemukan kesalahan", tempat sampah memori dipakai, penelusuran runtime dilakukan, program dibebani

dengan statement WRITE. (Kadang disebut juga dengan istilah *Delta Debugging*)



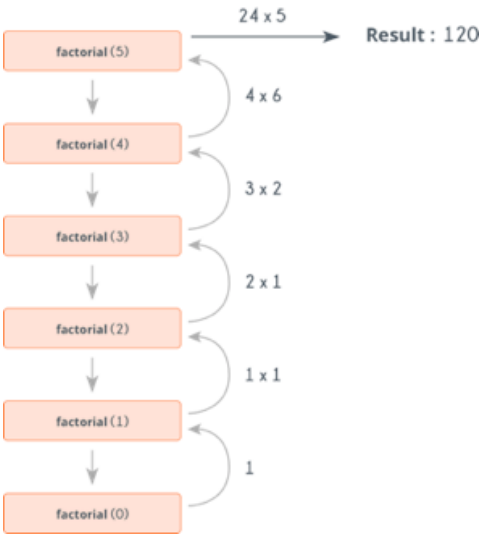
2. **Eliminasi penyebab** (*cause elimination*)

   Data yang berhubungan dengan kejadian kesalahan dikumpulkan. Hipotesis penyebab dibuat, data digunakan untuk membuktikan hipotesis diterima atau ditolak.

   Penjelasan lainnya, yaitu dengan cara mengumpulkan berbagai penyebab yang mungkin. Lalu dilakukan test case untuk mengeliminasi penyebab-penyebab yang tidak relevan. Setelah mengerucut pada satu penyebab yang mungkin, lalu dilakukan penghilangan bug di sekitar area tersebut.

3. **Penelusuran balik** (*backtracking*)

   Secara umum sukses untuk program kecil. Mulai pada sisi gejala diungkap, kode sumber ditelusuri balik (secara manual) sampai penyebab ditemukan.



1.9 Sebelum Dilakukan Koreksi BUG

Tiga pertanyaan sederhana yang harus diajukan kepada perekayasa perangkat lunak sebelum melakukan koreksi yang menghilangkan penyebab suatu BUG (Van Vleck (1989)):

1. Apakah penyebab bug direproduksi di dalam bagian lain program tersebut?

2. Apa "bug selanjutnya" yang akan dimunculkan oleh perbaikan yang akan dibuat?

3. Apa yang dapat kita lakukan untuk menghindari bug ini di dalam tempat pertama?

-ooOoo-