Kualitas Software dan Pengujian

Pendahuluan

Kualitas (dalam bahasa Inggris: *quality*, berasal dari bahasa latin: *qualitas*) merupakan konsep yang selalu dicari pada setiap apapun yang dibuat oleh manusia. Dari sisi filosofis, kualitas merupakan hal yang sulit untuk didefinisikan. Tentu saja kualitas ini tidak muncul khusus untuk software. Bisa dikatakan kualitas mempunyai umur yang sama dengan umur manusia diciptakan. Meskipun demikian, pembahasan untuk ini akan kita batasi pada kualitas produk yang dihasilkan oleh manusia, khususnya produk berupa software.

Arti dari Kualitas Software

Kualitas merupakan hal yang sangat kompleks, mempunyai arti yang lain dari satu orang ke orang lain, serta sangat bersifat kontekstual. Kitchenham dan Pfleeger mendefinisikan 5 pandangan yang berkaitan dengan kualitas software:

1. Pandangan Transcendental: bisa dirasakan tetapi sulit didefinisikan.
2. Pandangan Pemakai: kesesuaian dengan kegunaan tertentu, apakah software tersebut memuaskan kebutuhan dan harapan dari pemakai?
3. Pandangan Pemanufakturan: kesesuaian dengan spesifikasi
4. Pandangan Produk: kualitas dipandang sebagai sesuatu hal yang melekat pada karakteristik inherent dari produk
5. Pandangan Berbasis-Nilai: kualitas tergantung pada seberapa besar konsumen bersedia membayar untuk produk tersebut

Untuk mendefinisikan kualitas software, ada beberapa model yang dibuat oleh lembaga-lembaga yang berwenang. Model-model tersebut antara lain:

1. ISO 9126: fungsionalitas, reliabilitas, efisiensi, maintainability, portabilitas.
2. CMM: evaluasi proses pengembangan software ke dalam berbagai level (1 - 5).

Untuk mendapatkan kualitas, pengembang seringkali berhubungan dengan pengujian. Ada dua model yang berkaitan dengan pengujian software:

1. TPI (Test Process Improvement)
2. TMM (Test Mature Model)

Peran dari Pengujian

Fried dan Voas mendefinisikan pengujian software sebagai proses verifikasi dari *assessment* (penaksiran) dan perbaikan kualitas software. Penaksiran kualitas software pada dasarnya bisa dibagi menjadi dua:

1. Analisis Statis: tidak melibatkan eksekusi software dalam kondisi nyata, lebih menekankan pada peninjauan berbagai dokumen serta hasil dan analisis algoritma.
2. Analisis Dinamis: melibatkan eksekusi software dalam kondisi nyata.

Verifikasi dan Validasi

Verifikasi merupakan aktivitas untuk meng-evaluasi suatu software dengan menentukan apakah produk yang dihasilkan pada fase tersebut sesuai dengan requirements (persyaratan/kebutuhan) yang ditetapkan pada saat dimulai fase tersebut.
Validasi merupakan aktivitas untuk memastikan bahwa software tersebut sesuai dengan tujuan penggunaannya.
Verifikasi berhubungan dengan bagaimana membangun suatu software dengan benar, sedangkan validasi berhubungan dengan bagaimana membangun suatu software yang benar.

Software Failure

Software failure berhubungan dengan mata rantai fault -> error -> failure. Fault merupakan kondisi yang menyebabkan terjadinya error. Error merupakan keadaan sistem yang tidak sesuai / kesalahan. Failure merupakan kegagalan yang muncul karena perilaku eksternal dari sistem yang tidak sesuai dengan spesifikasi sistem.
Ram Chillarege mendefinisikan software failure sebagai 'harapan konsumen software

tidak bisa dipenuhi dan/atau konsumen tersebut tidak bisa mengerjakan hal yang bermanfaat dengan produk software tersebut'.

Software Reliability

Software reliability berkaitan dengan probabilitas dari operasi yang bebas-kegagalan dari suatu software pada suatu kurun waktu tertentu dan pada suatu lingkungan tertentu.

Pandangan-pandangan Terhadap Proses Pengujian

Berbagai stakeholders dari proses pengujian (pemrogram, test engineers, manajer proyek, konsumen) mempunyai berbagai pandangan yang berbeda-beda:

1. It does work
2. It doesn't work
3. Pengurangan resiko failure
4. Pengurangan biaya pengujian

Biaya pengujian meliputi:

1. Biaya merancang, memaintain, dan mengeksekusi test case
2. Biaya analisis hasil eksekusi test case
3. Biaya untuk mendokumentasikan test case
4. Biaya dari mengeksekusi sistem sesungguhnya dan mendokumentasikannya.

Test Case

Test Case merupakan pasangan dari <input, hasil yang diharapkan>

Konsep Complete Testing

Complete Testing merupakan hal yang tidak mungkin dilakukan karena berbagai keterbatasan pengujian. Selama ini jika membicarakan complete testing berarti 'tidak ada fault yang ditemukan sampai akhir dari fase pengujian'.

Aktivitas Pengujian

1. Identifikasi tujuan yang akan diuji
2. Pilih input
3. Tentukan hasil yang diharapkan berdasarkan input tersebut
4. Tentukan execution environment
5. Eksekusi program
6. Analisis hasil pengujian

Level-level Pengujian

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing (dilakukan oleh konsumen)

1 - 3 => kemungkinan terdapat regression testing (testing yang dilakukan setelah ada bagian yang dimodifikasi).

Sumber-sumber Test Case

1. Requirements dan Functional Specification
2. Source code
3. Input dan Output domain
4. Operational profile
5. Fault Model

White Box dan Black Box Testing

White Box (Structural) Testing: melihat source code dengan fokus pada aliran kendali dan aliran data.
Black Box (Functional) Testing: tidak melihat rincian internal dari sistem, hanya berkaitan dengan apakah fungsionalitas berjalan dengan baik atau tidak.

Perancangan dan Perencanaan Pengujian

Kegunaan dari test planning adalah utk menyiapkan dan terorganisasi untuk eksekusi test. Suatu test plan menyediakan rerangka, ruang lingkup, rincian sumber daya yang diperlukan, usaha yang diperlukan, jadwal aktivitas, dan anggaran.