PERTEMUAN KE-6
BLACK BOX TESTING

6.1 TUJUAN PEMBELAJARAN

Adapun tujuan pembelajaran yang akan dicapai sebagai berikut:

6.1. Memahami Black Box Testing

6.2. Memahami Dekomposisi Kebutuhan untuk Testing Sistematis

6.3. Memahami Graph Based Testing

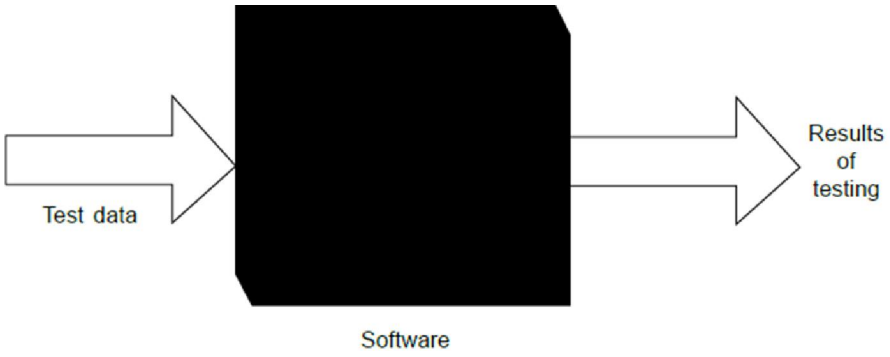
6.4. Memahami Equivalence Partitioning

6.5. Memahami Boundary Value Analysis

6.2 URAIAN MATERI

Tujuan Pembelajaran 6.1:
Black Box Testing

Pengujian black-box juga disebut uji fungsional, teknik pengujian yang mengabaikan detail internal sistem dan hanya berfokus pada masukan yang diterima, keluaran yang dihasilkan, dan kondisi eksekusi (Naik & Tripathy, 2008, p.163). Perangkat lunak diperlakukan sebagai kotak hitam, logika internal diabaikan, satu set masukan diberikan, kemudian keluaran dibandingkan dengan yang diharapkan, digambarkan sebagai berikut (Chemuturi, 2011, p.139):



Pengujian black box cenderung untuk menemukan berbagai jenis kesalahan:

- Fungsi yang hilang

- Masalah kegunaan
- Masalah kinerja
- Kesalahan waktu
- Inisialisasi dan penghentian kesalahan
- Dan lain-lain

Tidak seperti pengujian white box, pengujian black box cenderung diterapkan setelah proses pembangunan.

Berikut ini adalah langkah-langkah pengujian Black Box (Chemuturi, 2011, p.140):

1. Menyiapkan unit perangkat lunak berupa file executable, dan menginstalnya pada sistem pengujian
2. Menyiapkan data master yang dibutuhkan untuk pengujian
3. Mempelajari rencana pengujian dan memperhatikan tujuan pengujian
4. Mempelajari uji kasus yang didesain untuk pengujian
5. Menjalankan program baik dari command line, atau graphical user interface
6. Jalankan uji kasus dalam urutan tertentu, catat hasilnya untuk menentukan gagal, atau berhasil
7. Jika ragu terhadap hasilnya, kembalikan ke kondisi sebelum pengujian, kemudian lakukan pengujian ulang
8. Setelah semua uji kasus dieksekusi, atur review manajerial, dan hasilnya laporkan ke pencetus/penggagas permintaan pengujian
9. Memberikan klarifikasi kepada pencetus/penggagas permintaan pengujian, membantu pihak yang terlibat dalam memahami, dan menyelesaikan cacat
10. Sebagaimana disyaratkan, lakukan uji regresi untuk memastikan bahwa cacat tel
11. Jika pada uji regresi masih menemukan cacat, ulangi langkah 7 sampai 9

Kerangka pengujian black box dapat ditunjukkan seperti tabel berikut ini:

Identitas Pengujian	Deskripsi	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan (Berhasil/Gagal)

Tujuan Pembelajaran 6.2:

Dekomposisi Kebutuhan untuk Testing Sistematis

Kebanyakan tester, saat memulai proyek testing, akan menghadapi masalah untuk memutuskan test cases apa yang akan mereka eksekusi untuk melakukan tes sistem mereka. Untuk dapat membuat test cases yang efektif, harus dilakukan dekomposisi dari tugas-tugas testing suatu sistem ke aktivitas-aktivitas yang lebih kecil dan dapat dimanajemeni, hingga tercapai test case individual. Tentunya, dalam disain test case juga digunakan mekanisme untuk memastikan bahwa test case yang ada telah cukup mencakup semua aspek dari sistem.

Pendesainan test case dilakukan secara manual, Tidak ada alat bantu otomasi guna menentukan test cases yang dibutuhkan oleh sistem, karena tiap sistem berbeda, dan alat bantu tes tak dapat mengetahui aturan benar-salah dari suatu operasi. Disain tes membutuhkan pengalaman, penalaran dan intuisi dari seorang tester.

Spesifikasi sebagai tuntunan testing

Spesifikasi atau model sistem adalah titik awal dalam memulai disain tes. Spesifikasi atau model sistem dapat berupa spesifikasi fungsional, spesifikasi kinerja atau keamanan, spesifikasi skenario pengguna, atau spesifikasi berdasarkan pada resiko sistem. Spesifikasi menggambarkan kriteria yang digunakan untuk menentukan operasi yang benar atau dapat diterima, sebagai acuan pelaksanaan tes.

Banyak kasus, biasanya berhubungan dengan sistem lama, hanya terdapat sedikit atau bahkan tidak ada dokumentasi dari spesifikasi sistem. Dalam

hal ini sangat dibutuhkan peran dari pengguna akhir yang mengetahui sistem untuk diikutsertakan ke dalam disain tes, sebagai ganti dari dokumen spesifikasi sistem. Walaupun demikian, harus tetap ada dokumentasi spesifikasi, yang bisa saja dibuat dalam bentuk sederhana, yang berisi sekumpulan obyektifitas tes di level atas.

Dekomposisi obyektifitas tes

Disain tes berfokus pas tda spesifikasi komponen yang dites. Obyektifitas tes tingkat atas disusun berdasarkan pada spesifikasi komponen. Tiap obyektifitas tes ini untuk kemudian didekomposisikan ke dalam obyektifitas tes lainnya atau test cases menggunakan teknik disain tes.

Terdapat banyak jenis teknik disain tes yang dapat dipilih berdasarkan pada tipe testing yang akan digunakan, yaitu:

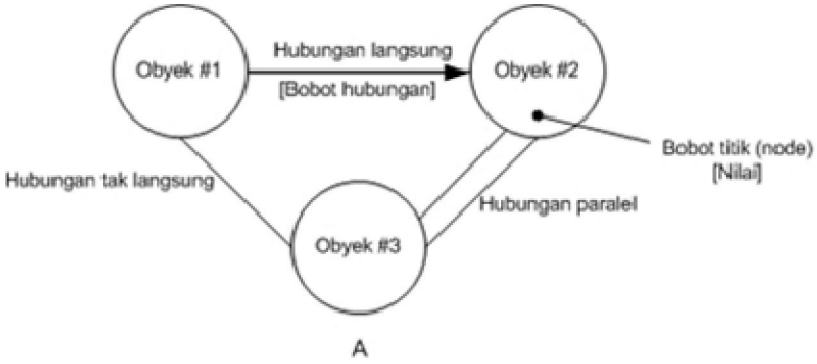
1. Equivalence Class Partitioning
2. Boundary Value Analysis
3. State Transitions Testing
4. Cause-Effect Graphing

Tujuan Pembelajaran 6.3:

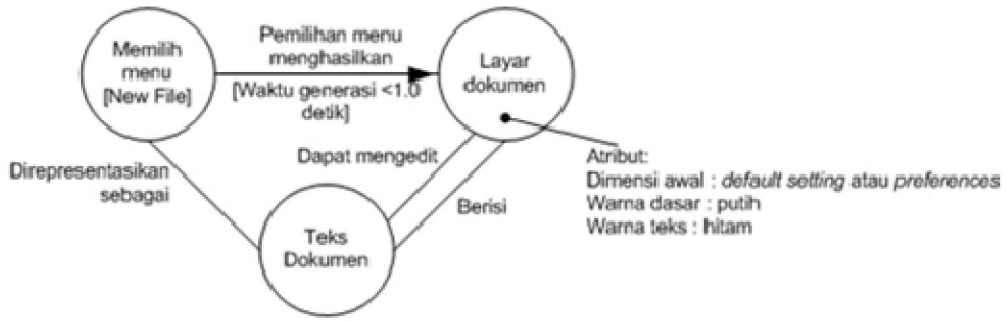
Graph Based Testing

Langkah pertama pada black box testing adalah memahami obyek yang dimodelkan dalam software dan hubungan koneksi antar obyek, kemudian definisikan serangkaian tes yang merupakan verifikasi bahwa semua obyek telah mempunyai hubungan dengan yang lainnya sesuai yang diharapkan.

Langkah ini dapat dicapai dengan membuat grafik, dimana berisi kumpulan node yang mewakili obyek, penghubung / link yang mewakili hubungan antar obyek, bobot node yang menjelaskan properti dari suatu obyek, dan bobot penghubung yang menjelaskan beberapa karakteristik dari penghubung / link.



A



B

Representasi secara simbolik dari grafik terlihat seperti pada gambar A. Nodes direpresentasikan sebagai lingkaran yang dihubungkan dengan garis penghubung. Suatu hubungan langsung (digambarkan dalam bentuk anak panah) mengindikasikan suatu hubungan yang bergerak hanya dalam satu arah. Hubungan dua arah, juga disebut sebagai hubungan simetris, menggambarkan hubungan yang dapat bergerak dalam dua arah. Hubungan paralel digunakan bila sejumlah hubungan ditetapkan antara dua nodes.

Tujuan Pembelajaran 6.4:

Equivalence Partitioning

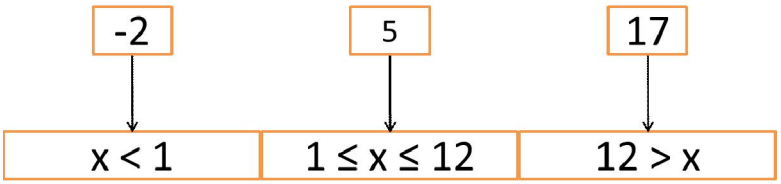
Equivalence partitioning adalah metode black box testing yang membagi domain masukan dari suatu program ke dalam kelas-kelas data, dimana test cases dapat diturunkan. Equivalence partitioning berdasarkan pada premis masukan dan keluaran dari suatu komponen yang dipartisi ke dalam kelas-kelas, menurut spesifikasi dari komponen tersebut, yang akan diperlakukan sama (ekuivalen) oleh

komponen tersebut. Dapat juga diasumsikan bahwa masukan yang sama akan menghasilkan respon yang sama pula.

Pada equivalence partitioning ruang/range masukan dibagi menjadi 2, valid dan tidak valid (Chemuturi, 2011, p.153).

Contoh Equivalence Partitioning:

Misalnya ada masukan berupa angka untuk bulan 1 sampai 12. Tentukan nilai valid dan tidak valid, kemudian pilih salah satu nilai dari masing-masing kelompok sebagai nilai yang mewakili.



Tujuan Pembelajaran 6.5:

Boundary Value Analysis

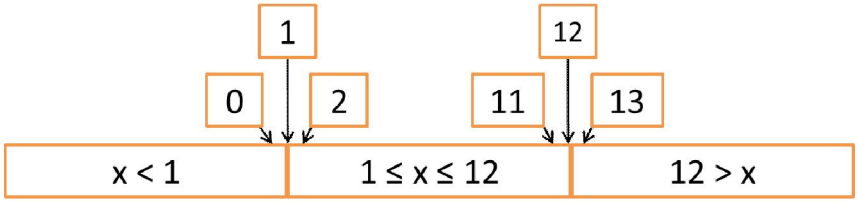
Ide utama dalam boundary value analysis adalah untuk memilih data uji yang dekat batas domain data sehingga data di dalam dan di luar yang dipilih memiliki kesetaraan kelas (Naik & Tripathy, 2008, p.246).

Untuk suatu alasan yang tidak dapat sepenuhnya dijelaskan, sebagian besar jumlah errors cenderung terjadi di sekitar batasan dari domain masukan daripada di “pusat”nya. Karena alasan inilah boundary value analysis (BVA) dikembangkan sebagai salah satu teknik testing. Boundary value analysis adalah

suatu teknik disain test cases yang berguna untuk melakukan pengujian terhadap nilai sekitar dari pusat domain masukan.

Teknik boundary value analysis merupakan komplemen dari teknik equivalence partitioning. Setelah dilakukan pemilihan tiap elemen suatu kelas ekuivalensi (menggunakan equivalence partitioning), BVA melakukan pemilihan nilai batas-batas dari kelas untuk test cases. BVA tidak hanya berfokus pada kondisi masukan, BVA membuat test cases dari domain keluaran juga.

Contoh BVA, untuk masukan berupa bulan 1 sampai 12, maka diambil nilai batas dan nilai yang mendekati batas seperti pada gambar berikut ini:



6.3 LATIHAN SOAL/TUGAS

1. Buatlah program sederhana untuk menghitung nilai akhir berdasarkan nilai UTS dan UAS. Kemudian lakukan pengujian black box menggunakan Equivalence Partitioning dan Boundary Value Analysis

6.4 DAFTAR PUSTAKA

1. Chemuturi, M. (2011). Mastering Software Quality Assurance. Best Practices, Tools And Techniques For Software Developers. J. Ross Publishing
2. Lewis, E. W. (2009). Software Testing and Continuous Quality Improvement. CRC Press
3. Naik, K. & Tripathy, P. (2008). Software Testing and Quality Assurance. Theory and Practice. John Wiley & Sons