WHITE BOX TESTING DAN BLACK BOX TESTING

White Box

[](http://2.bp.blogspot.com/-YGLIjNDzifE/UpxNdCK965I/AAAAAAAAAFU/XCxk_AxKK64/s1600/black+box.jpg)

*Pengertian White Box Testing*

White box testing 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. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Pengujian dilakukan berdasarkan bagaimana suatu software menghasilkan output dari input . Pengujian ini dilakukan berdasarkan kode program.

 Disebut juga struktural testing atau glass box testing

Teknik pengujian :

**1.      Menggambarkan kode program ke dalam graph yaitu node & edge**.

Jika berhubungan bernilai 1, bila tidak bernilai nol.

Dalam pengujian ini akan diperoleh hasil :

\* Kemungkinan source code yang dieksekusi

\* Waktu yang dibutuhkan

\* Memori yang digunakan

\* Sumber daya yang digunakan

**2.** **Basic path**, yaitu pengukuran kompleksitas kode program dan pendefinisian alur yang akan dieksekusi.

Digambarkan sequence, if, atau while nya

Uji coba basis path adalah teknik uji coba white box yg diusulkan Tom McCabe. Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunkan ukuran ini sbg petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. Test case yg didapat digunakan untuk mengerjakan basis set yg menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

**3.** **Data flow testing,** untuk mendeteksi penyalahgunaan data dalam sebuah program.

**4. Cyclomatic Complexity**  
Cyclomatic Complexity merupakan suatu sistem pengukuran yang menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Pada Basis Path Testing, hasil dari cyclomatic complexity digunakan untuk menentukan banyaknya independent paths. Independent path adalah sebuah kondisi pada program yang menghubungkan node awal dengan node akhir.

Terdapat 2 persamaan yang digunakan, yaitu:

V(G)= E - N + 2 atau V(G)= P + 1

Keterangan:

V(G)= cyclomatic complexity untuk flow graph G

E=Jumlah edge(panah)

N=Jumlah node(lingkaran)

P=Jumlah predicate node

o    *Kelebihan White Box Testing*

**Kesalahan logika.** Digunakan pada sintaks ‘if’ dan pengulangan. Dimana White Box Testing akan mendeteksi kondisi-kondisi yang tidak sesuai dan mendeteksi kapan proses pengulangan akan berhenti.

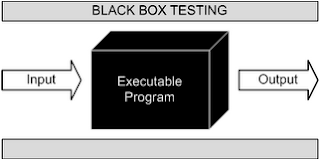
**Ketidaksesuaian asumsi.** Menampilkan asumsi yang tidak sesuai dengan kenyataan, untuk di analisa dan diperbaiki.

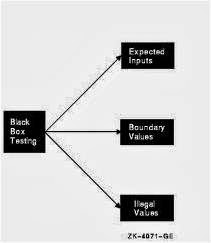
**Kesalahan ketik.**Mendeteksi bahasa pemrograman yang bersifat case sensitive.

o    *Kelemahan White Box Testing*

                      - Untuk perangkat lunak yang tergolong besar, White Box Testing dianggap sebagai strategi yang tergolong boros, karena akan melibatkan sumber daya yang besar untuk melakukannya

Black Box

[](http://4.bp.blogspot.com/-5rF2-HdVIyQ/UpxNfzFu_JI/AAAAAAAAAFc/P9zl3D1-or8/s1600/2.png)

[](http://4.bp.blogspot.com/-v7JjFnu_aUI/UpxNkpfr4EI/AAAAAAAAAFs/TJnzRnIPd1w/s1600/3.jpg)

*Pengertian Black Box Testing*

Black box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Jadi dianalogikan seperti kita melihat suatu koatak hitam, kit hanya bisa melihat penampilan luarnya saja, tanpa tau ada apa dibalik bungkus hitam nya. Sama seperti pengujian black box, mengevaluasi hanya dari tampilan luarnya(interface nya) , fungsionalitasnya.tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui input dan output).

Black Box pengujian adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian white-box). Pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan output yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit testing juga.

Pengujian pada Black Box berusaha menemukan kesalahan seperti:

* Fungsi-fungsi yang tidak benar atau hilang
* Kesalahan interface
* Kesalahan dalam struktur data atau akses database eksternal
* Kesalahan kinerja
* Inisialisasi dan kesalahan terminasi

Teknik khas Black Box Testing desain meliputi:

*1. DECISION TABLE*

Decision Tablel adalah cara yang tepat belum kompak untuk model logika rumit, seperti diagram alur dan jika-then-else dan switch-laporan kasus, kondisi mengaitkan dengan tindakan untuk melakukan, tetapi dalam banyak kasus melakukannya dengan cara yang lebih elegan.

Pada tahun 1960-an dan 1970-an berbagai “Decision Table Based“ bahasa seperti Filetab sangat populer untuk pemrograman bisnis.

*2. ALL-PAIRS TESTING*

All-pairs testing atau pairwise testing adalah metode pengujian perangkat lunak kombinatorial bahwa, untuk setiap pasangan parameter masukan ke sistem (biasanya, sebuah algoritma perangkat lunak), tes semua kombinasi yang mungkin diskrit parameter tersebut. Menggunakan vektor uji dipilih dengan cermat, hal ini dapat dilakukan jauh lebih cepat daripada pencarian lengkap semua kombinasi dari semua parameter, dengan “parallelizing“ pengujian pasangan parameter. Jumlah tes biasanya O (nm), dimana n dan m adalah jumlah kemungkinan untuk masing-masing dua parameter dengan pilihan yang paling.

Alasan di balik semua-All-pairs testing ini: yang sederhana dalam sebuah program umumnya dipicu oleh parameter masukan tunggal. Kategori paling sederhana berikutnya bug terdiri dari mereka bergantung pada interaksi antara pasangan parameter, yang bisa ditangkap dengan menguji semua-pasangan. yang melibatkan interaksi antara tiga atau lebih parameter secara progresif kurang umum [2], sementara pada saat yang sama waktu semakin lebih mahal untuk mencari oleh pengujian mendalam, yang sebagai batas pengujian lengkap semua input yang mungkin.

Banyak metode pengujian menganggap semua-pasang pengujian sistem atau subsistem sebagai kompromi biaya-manfaat yang wajar antara sering komputasi tidak layak tingkat tinggi metode pengujian kombinatorial, dan metode yang kurang lengkap yang gagal untuk menjalankan semua pasangan yang mungkin dari parameter. Karena tidak ada teknik pengujian dapat menemukan semua bug, semua-pasangan pengujian biasanya digunakan bersama dengan berbagai teknik jaminan mutu seperti unit testing, eksekusi simbolik, pengujian bulu halus, dan memeriksa kode.

*3. STATE TRANSITION TABLE*

Dalam teori automata dan logika sekuensial, state transition table adalah tabel yang menunjukkan apa yang negara (atau negara dalam kasus robot terbatas nondeterministic) suatu semiautomaton terbatas atau mesin finite state akan pindah ke, berdasarkan kondisi saat ini dan masukan lainnya. Sebuah tabel negara pada dasarnya adalah sebuah tabel kebenaran di mana beberapa input adalah kondisi saat ini, dan output termasuk negara berikutnya, bersama dengan keluaran lain.

state transition table adalah salah satu dari banyak cara untuk menentukan mesin negara, cara lain menjadi diagram negara, dan persamaan karakteristik.

*4. EQUIVALENCE PARTITIONING*

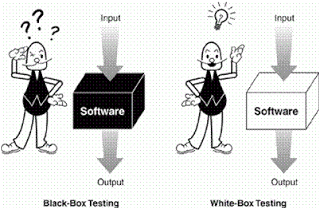
Equivalence partitioning adalah pengujian perangkat lunak teknik yang membagi data masukan dari unit perangkat lunak menjadi beberapa partisi data dari mana test case dapat diturunkan. Pada prinsipnya, uji kasus dirancang untuk menutupi setiap partisi minimal sekali. Teknik ini mencoba untuk mendefinisikan kasus uji yang mengungkap kelas kesalahan, sehingga mengurangi jumlah kasus uji yang harus dikembangkan.

Dalam kasus yang jarang Equivalence partitioning juga diterapkan pada output dari komponen perangkat lunak, biasanya itu diterapkan pada masukan dari komponen diuji. Partisi ekivalen biasanya berasal dari spesifikasi persyaratan untuk atribut masukan yang mempengaruhi pengolahan benda uji. Sebuah masukan telah rentang tertentu yang rentang sah dan lainnya yang tidak valid. Data yang tidak valid di sini tidak berarti bahwa data tidak benar, itu berarti bahwa data ini terletak diluar dari partisi tertentu. Hal ini mungkin lebih tepat dijelaskan oleh contoh fungsi yang mengambil sebuah parameter “bulan“. Jangkauan bulan adalah 1 sampai 12, mewakili Januari-Desember. Jangkauan ini disebut partisi. Dalam contoh ini ada dua partisi lebih lanjut rentang tidak valid. Partisi pertama akan menjadi tidak valid <= 0 dan partisi tidak valid kedua akan menjadi> = 13.

*5. BOUNDRY VALUES ANALYSIS*

Boundary value analysis merupakan suatu teknik pengujian perangkat lunak di mana tes dirancang untuk mencakup perwakilan dari nilai-nilai batas. Nilai-nilai di tepi sebuah partisi kesetaraan atau sebesar nilai terkecil di kedua sisi tepi. Nilai dapat berupa rentang masukan atau keluaran dari komponen perangkat lunak. Karena batas-batas tersebut adalah lokasi umum untuk kesalahan yang mengakibatkan kesalahan perangkat lunak mereka sering dilakukan dalam kasus-kasus uji.  
Dokumentasi komponen software, mencangkup pemeriksaan dokumen dari software itu sendiri, yaitu :  
\* Flowchart yang dibuat  
\* Deskripsi input yang digunakan  
\* Deskripsi output yang digunakan  
\* Deskripsi output yang dihasilkan  
\* Kesesuaian penulisan (akurasi)  
\* Kontrol/kendali terhadap sistem yang dibuat

Strategi Black Box System, meliputi :  
\***Batasan nilai untuk testing**, meliputi beberapa nilai, yaitu  
   - Nilai minimum variabel input  
   - Nilai di atas nilai minimum  
   - Nilai normal  
   - Nilai di bawah nilai maksimum  
   - Nilai maksimum  
\* **Equivalent Class Testing,**yaitu mengelompokkan input yang direpresentasikan sebagai hasil yang valid atau invalid. Contoh :  
Rekruitasi pegawai berdasarkan pengalaman kerja :  
<1thn    : diterima, part time  
1-3 thn  : diterima, sebagai tenaga kerja profesional  
>4 thn  : diterima, sebagai pegawai tetap  
  
Kesalahan yang dapat terdeteksi melalui testing ini ialah :  
\* kebenaran dokumentasi  
\* akses basis data  
\* hasil akhir program  
  
Kelebihan black box testing :  
\* Spesifikasi program dapat ditentukan di awal  
\* Dapat digunakan untuk menilai konsistensi program  
\* Testing dilakukan berdasarkan spesifikasi  
\* Tidak perlu melihat kode program secara detail  
  
Kekurangan black box testing :  
\* Bila spesifikasi program yang dibuat kurang jelas dan ringkas, maka akan sulit membuat dokumentasi setepat mungkin

[](http://4.bp.blogspot.com/-pgsZLVCrwzg/UpxNjULIMII/AAAAAAAAAFk/6anEcaq69sc/s1600/4.gif)

Perbedaan White Box & Black Box

o    *White box (Struktural)*

  Dilakukan oleh penguji yang mengetahui tentang QA.

  Melakukan testing pada software/program aplikasi menyangkut security dan performance program tersebut (meliputi tes code, desain implementasi, security, data flow, software failure).

  Dilakukan seiring dengan tahapan pengembangan software atau pada tahap testing.

o    *Metode BlackBox  (Fungsional)*

  Dilakukan oleh penguji Independent.

  Melakukan pengujian berdasarkan apa yang dilihat, hanya fokus terhadap fungsionalitas dan output. Pengujian lebih ditujukan pada desain software sesuai standar dan reaksi apabila terdapat celah-celah bug/vulnerabilitas pada program aplikasi tersebut setelah dilakukan white box testing.

  Dilakukan setelah white box testing.