PERTEMUAN 15
(UJICOBA BERORIENTASI OBJEK)

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat menerangkan Ujicoba Unit, Ujicoba Validasi, Ujicoba Integrasi dan Ujicoba Sistem.
2. Mahasiswa dapat memahami Ujicoba yang berorientasi dengan Objek.

B. URAIAN MATERI

1. Pengujian Perangkat Lunak

Pengujian pada sebuah software bisa dimulai untuk menentukan kebutuhan software yang terkait, kemudian memperoses dan berlanjut kebentuk desain, hingga ke pengkodeaan akhir. Ujicoba yang sama dengan hal tersebut, dapat di mulai dengan unit ujicoba di pusat spiral,setiap masing-masing modul atau unit dari software yang di implementasikan dalam *sourcecode* menjadi tujuan pengujian. Kemudian dilakukan pengujian integrasi, yang berpusat pada pengujian adalah desain kontruksi arsitektur perangkat lunak. lalu dilakukan validation pengujian dengan pusat sasaran pengujian adalah kesesuaian dengan persyaratan sebuah software yang telah ditentukan di awal. Berakhir di lingkaran luar spiral hingga pada sistem pengujian, dimana perangkat lunak dan kesemua sistem diuji.

a. Metode Pengujian Perangkat Lunak

Pengujian yang merupakan sebuah rangkaian kegiatan yang bisa direncanakan sebelum dikerjakan secara sistematis. Strategi ujicoba perangkat lunak memudahkan para perancang untuk menentukan keberhasilan sistem yang telah dikerjakan. Hal yang perlu diperhatikan ialah tahapan dan pelaksanaan yang di rencanakan dengan baik dan dengan waktu berapa lama, upaya dan sumber daya yang diperlukan ujicoba mempunyai karakteristik sebagai berikut :

1) Tes dimulai pada dari level terendah, dan di teruskan menggunakan modul diatasnya lalu hasilnya digabungkan.
2) Ada beberapa hal yang membedakan dari Teknik pengujian yang di lakukan (dalam hal waktu)

3) Pengembang melakukan pengujian perangkat lunak (untuk proyek yang besar) dalam suatu kelompok pengujian yang independent.
4) Aktivitas yang berbeda merupakan Pengujian dan debugging, tetapi debugging termaksud dalam startegi pengujian.

a) Validasi dan Verifikasi

Validasi dan Verifikasi merupakan tahapan pengujian perangkat lunak. Verifikasi berpacu pada rangkaian aktifitas agar dapat di pastikan bahwa perangkat lunak mengimplementasikan fungsi tertentu secara benar, sedangkan Validasi mengacu pada serangkaian kegiatan untuk dapat memastikan bahwa perangkat lunak yang diproduksi sesuai dengan kebutuhan konsumen. Definisi V&V meliputi serangkaian aktivitas dari *Software Quality Assurance* (SQA) yang meliputi kajian teknis formal, audit kualitas dan control, studi feabilitas, kajian dokumentasi, kajian basisdata, analisah logaritma, monitoring, kinerja, simulasi, pengujian kualifikasi, pengujian pengembangan, dan pengujian instalasi.

b) Pengorganisasian pengujian Perangkat Lunak

Perjalanan sebuah perangkat lunak yang di ujicoba sebaiknya melibatkan banyak ornag yang secara khusus dapat bertanggung jawab untuk melakukan peroses pengujian secara independent. Untuk itulah di perlukan *Independent Test Group* (ITG). Fungsi dari ITG adalah untuk menghilangkan "*conflict of interest*" saat terjadi Ketika pengembang sebuah software berusaha untuk menguji produknya pribadi. walaupun seperti itu, dapat terjadi beberapa kesalahan pemahaman yang terkait pada peran ITG, antara lain:

(1) Pengembang dapat melakukan tes sama sekali. Dan pendapat ini tidak 100% pasti benar, karena dalam beberapa kasus, pengembang pengmbang dapat melakukan pengujian dan integration test.

(2) Software akan di berikan begitu saja untuk diuji secara sporadic. Hal ini adalah suatu kesalahan karena pengembang dan ITG berkontibusi pada kesalahan proyek untuk memastikan pengujian akan di lakukan. Saat pengujian dikerjakan

pengembang diwajibkan memperbaiki kesalahan yang di temukan.

(3) Penguji tidak terkait pada proyek sampai tahap pengujian dimulai. Hal ini merupakan kesalahan karena ITG merupakan bagian dari tim proyek pengembangan dari software dimana mereka ada dalam spesifikasi proses dan tetap terlihat pada keseluruhan proses dan tetap terlihat pada keseluruhan proyek besar.

b. Masalah-masalah Strategis

1) Dapat di spesifikasikan bahwa kebutuhan produk pada sikap yang terukur sebelum pengujian dimulai. Susunan pengujian yang baik dapat menemukan kesalahan, tetapi juga untuk menentukan menilai kualitas program.
2) Tujuan yang di spesifikasikan terhadap pengujian secara eksperangkat lunakisit. Sasaran spesifik terhadap pengujian harus dinyatakan dalam bentuk yang terukur.
3) User perangkat lunak yang di identifikasi sebagai kategori untuk dan membuat profilnya, masing-masing kategori. Beberapa permasalahan dapat menggambarkan scenario interaksi bagi masing-masing kategori dapat mengurangi kinerja pengujian dengan berfokus pada pengujian penggunaan actual produk.
4) Mendirikan suatu rencana pengujian yang menegaskan rapid cycle testing. Umpan balik yang muncul dari rapid cycle testing yang dapat difungsikan untuk mengontrol kualitas dan strategi pengujian yang sesuai.
5) Membangun software yang kuat untuk merancang tes itu sendiri. Perangkat lunak juga mampu mendiagnosis berbagai jenis kesalahan tertentu dan beradaptasi dengan pengujian otomatis dan pengujian regresi.
6) Gunakan tinjauan formal yang efektif sebagai filter sebagai filter pra-tes. Kajian teknis formal yang dapat membuktikan sebuah kesalahan seefektif mungkin atas pengujiannya sehingga dapat mengurangi beban kerja pengujian.
7) Membuat sebuah tinjauan formal yang dapat mengungkapkan inkonsistensi, penghapusan, dan kesalahan dalam pendekatan pengujian.

8) Tetapkan metode pengekangan berkelanjutan untuk proses pengujian. Strategi pengujian harus dapat diukur. Metric yang dikumpulkan selama pengujian harus digunakan sebagai bagian dari metode pengendalian proses statistic pengujian perangkat lunak.

c. Pengujian Unit

Unit testing (Ujicoba Unit) berfokus dengan suatu usaha verifikasi pada unit yang lebih kecil dari desain software, yaitu modul. Ujicoba satuan yang berorientasi dengan *White-box testing* dan dapat dibuat secara pararel pada modul lainnya.

1) Pertimbangan pengujian Unit

Antarmuka modul untuk pengujian yang memastikan bahwa sebuah informasi dapat secara tepat masuk dan keluar dari pusat program yang diujikan. Pengujian struktur data local yang pengujiannya untuk memastikan dengan sebuah data agar dapat tersimpan secara temporal dan dapat menjamin integritasnya selama semua Langkah dalam algoritma dapat dieksekusi. Keadaan batas yang pengujiannya dapat memastikan modul berperan dengan tepat dengan suatu batas yang membatasi sebuah prosesnya. Seluruh jalur independent (jalur dasar) yang melewati struktur control digunakan sedikitnya 1 x (satu kali). Dan akhirnya penanganan kesalahan di uji.

2) Prosedur Pengujian Unit

Sumber telah dikembangkan, di tunjang Kembali dan di verifikasi untuk sintaks, maka perancangan *test-case* dimulai. Peninjauan kembali perancangan informasi akan menyediakan petunjuk untuk menentukan *testcase*.karena modul bukan program yang berdiri sendiri maka driver (pengendali) dan Stub perangkat lunak harus dikembangkan untuk pengujian unit. Driver adalah program yang menerima data untuk *testcase* dan menyalurkan ke modul yang diuji dan mencetak hasilnya.

d. Pengujian Integrasi

Testing terintegrasi sebuah cara yang sistematis untuk penyusunan struktur program, saat dilakukan ujicoba yang akan memeriksa kesalahan nantinya akan digabungkan melalui *interface.* Metode pengujian yaitu :

1) *Top Down Integration*

Ialah metode inkrmental yang pengembangan struktur program. Modul terintegrasi dengan memulainya dari modul utama dan turun kehierarki kontrol. Modul di bawah modul control utama yang di gabungkan kedalam struktur secara mendalam-pertama *(depth first) t*atau luas-pertama *(breadth first)*. Proses integrasi :

a) Modul utama yang di gunakan sebagai test driver dan stub yang digantikan semua modul dengan langsung dibawah modul control pusat.
b) bergantung pada pendekatan perpaduan yang di pilih (depth / breadth)
c) Ujicoba dapat dilakukan selama masing-masing modul di padukan
d) Masing-masing Penyelesaian ujicoba stub yang lain akan di pindahkan dengan modul yang sebenarnya
e) Ujicoba regression ialah pengujian yang diulang untuk mendapatkan kesalahan lain yang kemungkinan akan muncul.

2) *Buttom up integration*

Ujicoba buttom up di nyatakan dengan susunan yang di mulai dan dicoba dengan *atomic modul* (modul tingkat paling bawah pada struktur program). Sebab modul yang dipadukan bawah ke atas, proses yang di perlukan untuk modul subordiat yang selalu diberikan harus ada dan diperlukan untuk stub yang akan dihilangkan. Strategi pengujian :

a) Modul level bawah di gabungkan kedalam cluster yang melihatkan subfungsi pada software.
b) Input testcase dan output yang diatur oleh Driver (program kontrol pengujian).
c) Cluster diuji
d) Driver di ganti dan cluster yang di kombinasikan di pindahkan ke atas oleh struktur program

e. Pengujian Validasi

Dengan semua permasalahan yang di perbaiki maka proses selanjutnya adalah *validation testing*. Ujicoba sebuah validasi dapat dikatakan berhasil bila perangkat lunak berfungsi sesuai dengan apa yang diharapkan user. Validasi

perangkat lunak ialah sekumpulan seri ujicoba kotak-hitam yang menunjukan tepat dengan apa yang diperlukan. Beberapa keadaan setelah ujicoba :

1) Karakter performansi dengan kesesuaian fungsi spesifikasi dan dapat diterima
2) Di temukannya sebuah penyimpangan dari spesifikasi yang akan dibuat sebuah daftar penyimpangan.

f. Pengujian BETA dan ALPA

Apabila sebuah softwareyang akan di buat untuk user hal dapat dilakukan ialah *Aceeptance Test* yang dapat memungkinkan user untuk memvalidasi semua keperluan. Pengujian ini dapat dilakukan karena memungkinkan seseorang menemukan pada sebuah kesalahan yang lebih detail dan hal ini membiasakan seseorang untuk memahami sebuah software yang telah di buat.

1) Pengujian BETA

Pengujian yang dilakukan kepada 1 (satu) user atau lebih pemakaian akhir pada sebuah software dalam lingkungan yang sebetulnya, pengujian ini bisanya tidak dihadirkan pengembang dan semua masalah di rekam oleh pelanggan (real atau imajiner) hal ini dapat di temui selama pengujian dan melaporkan pada pengembang pada interval waktu tertentu.

2) Ujicoba ALPHA

Dari sisi pengmbang yang melakukannyaterhadap seorang pengguna. Software dengan pengembang “ yang memandang “ melalui bahu pemakai dan merekam semua kesalahan dan masalah pemakaian.

g. Pengujian Sistem

Akhirnya, software yang disatukan dengan element sistem yang lain, dan dan serangkaian integrasi sistem dan pengujian validasi dilakukan. Bila ujicoba gagal atau melampaui ruang lingkup ruang lingkup pengembang sistem, Langkah-langkah yang diambil selama desain dan pengujian dapat diperbaiki. Kesuksesan untuk perpaduan sebuah software dan sistem yang

besar adalah kuncinya. Pengujian system adalah serangkaian pengujian yang berbeda, tujuan utamanya adalah bekerja pada semua element sistem yang dikembangkan, yaitu ;

1) Tes pemulihan *(Recovery Testing)*

   Tes pemulihan Adalah system pengujian yang memaksa sebuah software mendapatkan suatu kegagalan dalam berbagai cara dan apakah perbaikan dilakukan dengan tepat.

2) Strees Testing

   Dibuat untuk menghadapi situasi abnormal selama pengujian program. Pengujian dapat dilakukan oleh system dalam menanggapi suatu kondisi seperti volume data yang abnormal (lebih besar dari atau kurang dari batasan) atau frekuensi.

h. Debugging

Debugging bukan lah suatu tes/pengujian, tetapi hasil dari suatu pengujian yang berhasil. Jika *testcase* menumukan *error*, maka proses debugging bertujuan untuk menghilangkan *error* tersebut. Debugging adalah proses yang sulit karena adanya bebearapa kesalahan umum, seperti :

1) Penyebab dari suatu kesalahan atau gejalanya mungkin saja akan jauh, karena gejala dapat muncul oleh bagian tertentu dari program dan penyebabnya bisa terdapat pada lokasi lain yang sangat jauh dari tempat timbulnya gejala.
2) Saat error dapat diperbaiki maka gejala akan hilang.
3) Gejala mungkin akan hadir pada sesuatu yang tidak salah, seperti pembulatan yang tidak tepat.
4) Gejala memungkinkan akan timbul dikarenakan masalah waktu.
5) Mungkin sulit secara akurat menghasilkan kondisi input.
6) Gejala dapat terjadi secara tiba-tiba.
7) Gejala dapat di sebabkan oleh sesuatu yang didistribusikan melewati sejumlah tugas yang bekerja pada prosesor yang berbeda-beda.

Terdapat 3 jenis pendekatan debugging, antara lain :

1) Kekuatan kasar (Brute force)

Ini adalah Teknik yang paling umum dipakai tetapi tidak paling efisien untuk mengisolasi penyebab kesalahan. Sejalan dengan prinsip " Biarkan Komputer menemukan error " oleh sebab itu menggunakan semua sumber daya komputer dengan tujuan untuk mengetahui penyebab kesalahan.

2) Pelacakan Kembali (Back Tracking)

Ini adalah sebuah metode pendekatan yang dimulai dari ditemukannya suatu gejala lalau ditelusuri Kembali penyebabnya.

3) Penyebab Eliminasi (Cause Elimination)

Dinyatakan melalui induksi atau deduksi,dan menggunakan konsep pembagian biner. Ialah kumpulan sebuah data yang terkait kesalahan untuk mengisolasi penyebabnya. Lalu akan dibuat sebuah hipotesis. Susunan penyebab yang kemungkinan dibuat dan dikerjakan oleh penguji untuk mengimplementasi penyebab-penyebab tersebut. Namun,bila pengujian menghasilkan kebenaran hipotesis pada suatu penyebab, maka data akan dikoreksi untuk mengisolasi kesalahan, dan bila kesalahan telah di temukan maka kesalahan harus di perbaiki. Namun, perbaikan pada kesalaham dapat menyebabkan kesalahan lain, jadi ada beberapa Tindakan pencegahan.

2. Object Oriented Test

Untuk *Object Oriented Test*, untuk unit terkecilnya ialah *object* & *class*, sistem merupakan sekelompok komunikasi antar *object*. Dan *class* selalu dirancang agar dapat diterima dengan pengurutan dipesan tertentu yang yang mencangkup tanggapan *class* pada pesa teks tersebut dan banyak perbedaan yang biasa dibilang dengan prilaku (*behavior*) class. Prilaku ini dikemudikan dengan nilai yang mengenkapsulasi, urutan pesan, atau keduanya.

a. Object Oriented Software Testing

Ujicoba software yang berorientasi pada sebuah objek ialah suatu proses yang dieksekusi progam untuk ditujukan dan ditemukannya kesalahan (Berard, 1994). Di temukannya Kesalahan yang bukan disengaja pada sebuah percobaan ini, apabila mampu memperbaiki kesalahan ini, maka pengujian dapat dianggap sukses. Maka dari itu, uji coba dapat berutujuan untuk

menunjukan penerapan fungsi pada sebuah software dan spesifikasi, ujicoba dapat dibagi menjadi beberapa kategori berikut :

1) Ujicoba pada proses pengembangan sistem dan dokumen pendukung. Proses mengacu pada banyaknya suatu aktivitas yang didukung oleh dokumen yang menjelaskan banyaknya aktivitas.
2) Ujicoba pada analisis dan desain analisis terhadap perancangan adalah sesuatu yang sangat penting.
3) Static dan dinamik pengujian ini untuk di implementasikan bertujuan untuk mencari kesalahan sebelumnya dalam suatu proses, namun kesalahan pada kode dalam program besar dan kompleks tidak bisa dihindarikan. Pengujian statis adalah pemeriksaan kode agar ditemukannya kegagalan logika. Ujicoba dinamis adalah data uji yang di eksekusi untuk menemukan kesalahan pada kode. *Software Oriented Object* berbeda dari *Software procedural* (konvensional) pada hal analisis, desain, teknik dan struktur pengembangan. Bahasa pemprograman berorientasi objek dicirikan dengan terdapat pengemasan *(encapsulation)*, anekaragam *(polymorphism)*. Dan warisan *(inheritance)* yang membutukan support beberapa pengujian. Berpusat pada ujicoba sebuat *software oriented object* di awali dengan hasil analisis dan desain yang harus dicek kembali dan yang utama dalam hal :
   a) Kebenaran sematik *(Semantic correctness)*, adlah penerapan model dan domain dalam masalah yang sebenarnya, apabila model mencerminkan keakuratan dalam dunia nyata, maka dapat ditentukan benar secara system matis,
   b) Konsistensi *(Consistency)*, ialah penerapan kelas dan objek turunannya, dan penerapan kelas dalam hubungan dengan kelas lainnya.

b. Test Model

Para peneliti telah mengusulkan berbagai model pengujian perangkat lunak berorientasi objek. Yang setiap modelnya memiliki struktur atau aturan yang menjadi dasar prosedur pengujian.

1) Model transisi keadaan

Dalam model pengujian, metode kelas (operasi) menunjukkan beberapa elemen perilaku di seluruh kelas. Beberapa metode desain berorientasi objek. Model transisi keadaan digunakan untuk mewakili

perilaku kelas. Keadaan suatu objek adalah kombinasi dari semua nilai atribut. Pada titik tertentu, keadaan statis. Dalam model objek dinamis, Anda perlu menambahkan transisi dari satu keadaan ke keadaan lain untuk melakukan tindakan. "Model status transisi" ditampilkan secara grafis, dengan simpul mewakili status dan busur mewakili transisi. Boris Beizer memberikan beberapa aturan untuk memeriksa model "transisi keadaan"yaitu:

a) Verifikasi bahwa status telah merepsentasikan himpunan dengan benar;
b) Cek model untuk semua kemungkinan event suatu class, yaitu transisi dari status untuk setiap metode berisi class dan pengecekan spesifikasi perancangannya;
c) pemeriksaan ketetapan satu transisi untuk masing-masing kombinasi “eventstate”. melakukan pengecekan, missal dengan representasi matriks sebagai kombinasi kemungkinan status dan event.
d) pengecekan yang tidak terjangkau, di mana pada status tersebut tidak ada lintasan atau perpindahan.
e) Cek aksi yang tidak benar (invalid), termasud method (operasi) yang tidak ada atau method yang tidak sesuai dengan kebutuhan selama transisi.

Model transisi-status dapat mencangkup keseluruhan prilaku class, sehingga lebih produktif di bandingkan dengan model lainya, namun mempunyai beberapa kelemahan yaitu:

a) Karena model dibentuk dari spesifikasi kebutuhan, bukan dari kode,mudah menyebabkan kesalahan sehingga perlu menguji model seperti halnya menguji kode
b) Karena model mencakup seluruh prilaku class dan superclass-nya maka model menjadi kompleks. Namun pemakaian “transisi-status” secara hararki dapat mengurangi kompleksitas tersebut
c) Model perilaku dapat mengakibatkan hilangnya kendali dan data yang salah. Olehkarena itu perlu adanya asumsi yang sama tentang perilaku class yang di dasarkan pada kebutuhan dengan asumsi yang dibuat oleh pemrogram.

c. Object Oriented Analysis (OOA) dan Object Oriented Design (OOD).
   1) Object Oriented Analysis (OOA)

Ini adalah metode analitik yang memeriksa persyaratan dalam hal kelas dan objek yang terjadi di domain masalah.

a) Menentukan persyaratan sistem melalui skenario atau kasus penggunaan.
b) Selanjutnya, buat model objek dengan kemampuan untuk memenuhi kebutuhan Anda.
c) Output : Model persyaratan yang biasanya menggunakan kartu CRC.
d) Memberikan penjelasan terperinci tentang sistem.
e) Identifikasi kebutuhan fungsional "apa" Use case) Identifikasi objek, kelas, operasi, atau hubungan objek, interaksi objek.
f) Buat model nyata menggunakan tampilan OO. Tujuan OOA adalah untuk memahami domain masalah dan meningkatkan akurasi, konsistensi, dan integritas.

2) Object-oriented Design

Desain berorientasi objek adalah tahapan perantara untuk menetapkan sepesifikasi atau kebutuhan system yang dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientsi objek.

a) Mendeskripsikan detail yang ada seperti "HOW", definisi kelas, kategori kelas, subsistem, arsitektur sistem.
b) OOA + detail implementasi.
c) Maintainability, reusability, extensibility, reliability dapat dioptimalkan sebagai tujuan untuk desain berorientasi objek.

3) Object-Oriented Testing

Proses pengujian sistem berorientasi objek, dimulai dengan tinjauan analisis dan model desain berorientasi objek (object-oriented analysis and design model). Setelah program dibuat, pengujian berorientasi objek (OOT) dimulai dengan pengujian "kecil" menggunakan pengujian kelas (manipulasi dan kolaborasi kelas). Ketika kelas-kelas ini diintegrasikan ke dalam subsistem, masalah dengan antar kelas menjadi diketahui. Terakhir, gunakan kasus penggunaan model OOA untuk menemukan kesalahan validasi perangkat lunak. OOT mirip dengan pengujian perangkat lunak tradisional dalam kasus uji dibuat untuk melatih kelas yang ada serta kolaborasi dan perilaku kelas. OOT

cenderung fokus pada masalah integrasi daripada pengujian unit. OOT mirip dengan pengujian perangkat lunak tradisional dalam kasus uji yang dibuat untuk melatih kelas yang ada dan bagaimana kelas bekerja sama. OOT cenderung fokus pada masalah integrasi daripada pengujian unit.

4) Aktivitas Pengujian Berorientasi Objek
   a) Memeriksa model OOA dan OOD
   b) Menguji kelas setelah menulis program sumber
   c) Menguji integrasi ke dalam subsistem
   d) Menguji integrasi subsistem yang ditambahkan ke sistem
   e) Uji Validasi berdasarkan OOA dan OOD
5) Pengujian Model OOA dan OOD
   a) OOA dan OOD tidak dapat diuji, tetapi akurasi dan konsistensi dapat diperiksa
   b) Konsistensi OOA dan OOD
   c) Konsistensi model OOA dan OOD
   d) Diagram model sistem (classresponsibilitycollaborator.) Dan hubungan antar objek .
   e) Memeriksa desain sistem (memeriksa model perilaku objek untuk memeriksa penugasan perilaku sistem ke subsistem, memeriksa pemrosesan paralel dan penetapan tugas, menggunakan skenario aplikasi untuk memeriksa desain antarmuka pengguna)
   f) Model Objek Uji Objek untuk Hubungan Menggunakan jaringan untuk memastikan bahwa setiap desain objek berisi atribut dan operasi yang diperlukan untuk melakukan kolaborasi yang ditentukan pada setiap kartu CRC.
   g) Tinjau spesifikasi terperinci dari algoritme yang digunakan untuk melakukan operasi tradisional menggunakan teknik inspeksi

d. Strategi Pengujian Berorientasi Objek

Pengujian Unit dalam Konteks Berorientasi Objek Unit terkecil yang diuji adalah enkapsulasi kelas atau objek, seperti pengujian sistem dalam perangkat lunak tradisional, dan dilakukan oleh orang lain. Jangan menguji operasi secara terpisah dari operasi dari. Pengujian lengkap kelas operasi dan perilaku, detail non-algoritmik, dan seluruh kelas aliran data melalui antarmuka antar modul meliputi:

1) Uji semua operasi yang terkait dengan objek

2) Atur dan integrasikan semua atribut objek
3) Latih objek dengan semua cara yang mungkin
4) Desain pengujian kelas menggunakan metode yang benar
   a) Pengujian yang berdasarkan kesalahan *(fault-based testing)*
   b) Pengujian di lakukan secara acak *(random testing)*

1) Pengujian Partisi *(Partition Testing)*
   a) Salah satu dari ini metode melakukan operasi yang dienkapsulasi oleh kelas.
   b) Prosedur pengujian dimaksudkan untuk memastikan bahwa proses yang relevan telah diuji.
   c) Tes untuk kesalahan posisi tetap (nilai atributnya) di kelas
2) Pengujian Integrasi dalam Konteks Berorientasi Objek
   a) Salah satu dari ini metode melakukan operasi yang dienkapsulasi oleh kelas.
   b) Prosedur pengujian dimaksudkan untuk memastikan bahwa proses yang relevan telah diuji.
   c) Tes untuk kesalahan posisi tetap (nilai atributnya) di kelas
   d) Pengujian berbasis kegunaan yang dimulai dengan pengujian independent oleh kelas pertama dan kelas-kelas yang tergantung pada penggunaannya.
   e) Pengujian Cluster (kerja sama kelompok kelas yang diuji untuk interaksi kesalahan)
   f) Pengujian regresi adalah pengujian yang penti karena setiap thread, cluster, atau subsistem yang ditambahkan pada suatu sistem.tingkat integrasi yang lebih sedikit berbeda dalam sistem berorientasi objek.
3) Validasi testing dalam konteks OO
   a) Berfokus pada Tindakan pengguna yang terlihat dan pengguna dapat mengenali output dari sistem
   b) Tes validasi didasarkan pada scenario use-case,model perilaku objek,dan diagram alur event dibuat dalam model OOA
   c) Pengujian black-box konvensional dapat digunakan untuk mendorong tes validasi.

4) Test Case Desain untuk software OO
   a) Setiap kasus uji harus dapat diidentifikasikan secara unit dan secara eksplisit dihubungkan dengan class yang akan diujikan
   b) Ditetapkan kegunaan dari setiap ujicoba
   c) Dapat dituliskan Langkah-langkah ujicoba untuk setiap pengujian yang diseratakan diantaranya yaitu:
      (1) Dapat dituliskan tahapan ujicoba untuk setiap objek yang disertakan dalam ujicoba
      (2) Pesan-pesan dan operasi yang di jalankan sebagai konsekuensinya dari ujicoba ini
      (3) Eksepsi yang muncul Ketika suatu objek di ujicoba.
      (4) Kondisi eksternal yang memerlukan perubahan untuk di ujicoba tersebut.
      (5) Dan informasi tambahan lainya yang di perlukan untuk memahami atau mengimplementasikan ujicoba tersebut.

e. Ujicoba struktur permukaan dan struktur dalam (*Testing Surface Structure and Deep structure)*

Pengujian struktur permukaan d. NS. Struktur pelatihan ditampilkan kepada pengguna akhir. Seringkali, Anda mengamati dan mengajukan pertanyaan saat bekerja dengan objek sistem. Tes struktur internal digunakan untuk melatih struktur program internal seperti dependensi, perilaku, dan mekanisme komunikasi yang ada sebagai bagian dari sistem dan desain objek. adalah metode tes yang dapat diterapkan di tingkat kelas.

Metode pengujian yang dapat di aplikasikan pada tingkat kelas yaitu.

1) Tes acak memerlukan sejumlah besar permutasi dan kombinasi data, yang bisa jadi tidak efisien.
2) Identifikasi kemungkinan operasi di kelas
3) Definisi batas aplikasi
4) Identifikasi urutan pengujian minimum dan beberapa variasi urutan pengujian acak

1) Uji Partisi
   a) Uji partisi menghilangkan jumlah kasus uji yang diperlukan untuk menguji suatu kelas.

b) Partisi berbasis status adalah pengujian yang dirancang untuk menguji operasi yang menyebabkan perubahan status, selain dari operasi yang tidak menyebabkan perubahan status.

c) Partisi berbasis atribut untuk setiap atribut kelas. Operasi dikategorikan berdasarkan pengguna atribut yang memodifikasi atribut dan tidak menggunakan atau memodifikasi atribut.

d) Partisi berbasis kategori adalah sistem operasi yang termasuk dalam basis. dengan fungsi untuk melakukan inisialisasi, kalkulasi, query, terminasi, dll

2) Fault-based testing

a) Paling cocok untuk tingkat operasional dan kelas.

b) Penggunaan struktur pewarisan.

c) Pengujian menguji model OOA untuk mengidentifikasi jumlah kerusakan yang dapat dipahami dalam panggilan operasional dan tautan pesan, dan kasus uji yang sesuai Buat.

d) Temukan spesifikasi. Ketidakakuratan dan kesalahan dalam interaksi subsistem

3) Inter-class Test Case Design

a) Desain kasus uji menjadi lebih rumit seperti halnya integrasi dari dimulainya sistem OO - menguji kolaborasi antar class

b) Ujicoba clas yang beragam seperti :

(1) Untuk setiap class client menggunakan daftar operator class untuk men-generate urutan pengujian secara acak mengirimkan pesan ke server lainnya.

(2) Untuk pesan yang di saring, tentukan class kolaborasi dan operator server object yang ditunjuk.

(3) Untuk masing-masing operator server class memilih pesan yang dikirimkan.

4) Pengujian yang di hasilkan dari Tipe perilaku

a) Gunakan diagram transisi status (STD) untuk model berbeda yang mewakili perilaku dinamis kelas.

b) Kasus uji harus mencakup semua tingkat STD.

c) Anda dapat menggunakan model traverse pertama dengan lebar rentang status (menguji transisi pada saat yang sama dan hanya menggunakan transisi yang telah diuji sebelumnya saat menguji transisi baru).
d) Seluruh perilaku kelas diuji dengan benar Anda dapat membuat kasus uji untuk memastikannya

5) Metode Pengujian Tingkat Antar Kelas.
   a) Fokus pada pengintegrasian dan pengujian kelompok objek yang bekerja sama.
   b) Memanipulasi objek dan grupnya Menggunakan pengetahuan Anda tentang penggunaan sistem untuk mengidentifikasi grup.
   c) Pendekatan pengujian klaster adalah kasus penggunaan atau skenario pengujian yang menggunakan pengujian berdasarkan interaksi pengguna dengan sistem dan memiliki keunggulan bahwa kemampuan pengujian sistem adalah kesadaran pengguna. Tes utas untuk menguji respons sistem terhadap peristiwa selain pemrosesan utas oleh sistem. Pengujian interaksi objek juga menguji urutan interaksi objek yang berhenti ketika operasi objek tidak memanggil layanan objek lain.
6) Pengujian Berbasis Usecase / Skenario
   a) Berdasarkan use case dan diagram urutan yang sesuai.
   b) Identifikasi skenario use case dan tambahkan diagram interaksi ke skenario yang menunjukkan objek yang dirujuk dalam skenario.
   c) Lakukan pengujian rutin setiap perilaku, dengan fokus pada persyaratan (fungsional) setiap kasus penggunaan, setiap ekstensi penuh gabungan (& lt;>) atau setiap ekstensi penuh gabungan (& lt;>).
   d) Proses adalah jalur melalui diagram urutan.
   e) Banyak skenario yang berbeda dapat digabungkan dengan diagram urutan.
   f) Digunakan untuk tugas pengguna dan variasinya dijelaskan dalam kasus penggunaan yang mempraktikkan tugas tersebut
   g) Menemukan perbedaan yang terjadi saat pengguna berinteraksi dengan perangkat lunak berorientasi objek .
   h) Produk Fokus pada fitur, bukan fitur.

i) Dapat membawa manfaat besar bagi tenaga dan waktu.

7) Masalah desain pengujian berorientasi objek

Metode pengujian kotak putih dapat digunakan untuk menguji kegunaan program untuk mengimplementasikan operasi kelas, tetapi tidak untuk pengujian lainnya. Metode pengujian kotak hitam sangat cocok untuk menguji pemrograman berorientasi objek dan berorientasi objek. Ini meningkatkan kebutuhan untuk pengujian. Artinya, terlihat seperti ini:

a) Kelas berisi operasi yang diturunkan dari superclass.
b) Subkelas dapat berisi operasi yang baru didefinisikan sebagai berikut.
c) Setiap kelas dihasilkan oleh tes dasar. Ini karena Anda dapat melanjutkan ke tes berikutnya sebelum itu.

C. SOAL LATIHAN/TUGAS

1. Sebutkan proses pada Validasi dan Verifikasi pengujian perangkat lunak ?
2. Ada berapa proses pada metode pengujian perangkat lunak ? minimal 3 !
3. Jelaskan tahapan dari Object Oriented Test ?
4. Jelaskan apa yang dimaksud dengan encapsulation pada uji coba berorientasi objek ?
5. Apa yang dimaksud dengan Model status transisi ?

D. REFERENSI

1. Jurnal Universitas Paramadina,Vol. 2, No. 2, Januari 2003 Oleh Retno Hendrowati,Dengan Judul Pengujian Perangkat Lunak Berorientasi Obyek
   Susilo, J. (2014). Aplikasi Pengujian White Box IBII Online Jugde. *Jurnal Informatika dan Bisnis*, *3*
2. 56-69.Jaya, T. S. (2018). Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung). *Jurnal Informatika: Jurnal Pengembangan IT*, *3*
3. 45-48.Anggawirya E dan Wit. (2001). Microsoft Windows 2000 Professional. Jakarta: Ercontara Rajawali.
4. Bambang Hariyanto. 1997. Sistem Operasi, Bandung:Informatika Bandung.

5. Dali S. Naga. 1992. Teori dan Soal Sistem Operasi Komputer, Jakarta: Gunadarma.
6. Ayuliana / testing dan implementasi / feb 2011 *Software Testing Strategis*.
   http://www.ofnisystems.com/services/validation/validation-master-plans/
   http://www.ofnisystems.com/services/validation/computer-systems/
   http://se.ittelkom-pwt.ac.id/software-testing-dalam-lingkup-software-engineering/
   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.261.1758&rep=rep1&type=pdf (Journal)
   http://www.scholarism.net/FullText/ijmcs20154303.pdf (Journal)
   https://www.researchgate.net/publication/270554162_A_Comparative_Study_of_White_Box_Black_Box_and_Grey_Box_Testing_Techniques (Journal)

GLOSARIUM

ITG (*Test Group*) berfungsi untuk menghilangkan "*conflict of interest*" saat terjadi Ketika pengembang sebuah software berusaha untuk menguji produknya pribadi.

Software Quality Assurance (SQA) yang meliputi kajian teknis formal, audit kualitas dan control, studi feabilitas, kajian dokumentasi, kajian basisdata, analisah logaritma, monitoring, kinerja, simulasi, pengujian kualifikasi, pengujian pengembangan, dan pengujian instalasi.

Object Oriented Analysis (OOA) Ini adalah metode analitik yang memeriksa persyaratan dalam hal kelas dan objek yang terjadi di domain masalah.