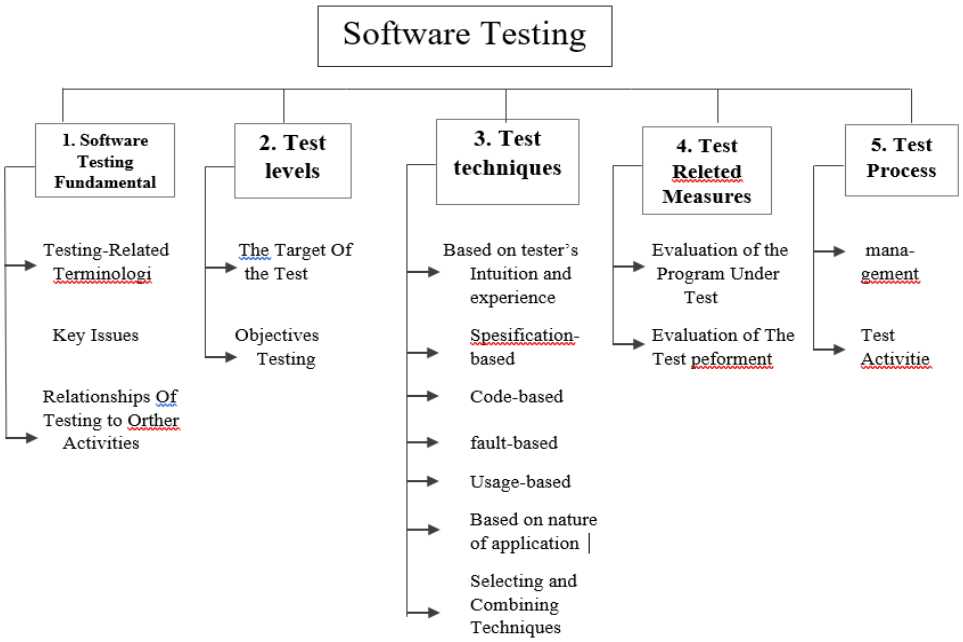
PERTEMUAN 14
(RENCANA UJICOBA VALIDASI SOFWARE)

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami Pembuatan rencana ujicoba
2. mahasiswa da pat memahami Teknik ujicoba black box dan white box

B. URAIAN MATERI

1. Pembuatan Rencana Ujicoba dan Studi kasus



Gambar 14. 1 Rincian Topik pada knowledge area software testing

Pengujian adalah perjalanan sistem yang digunakan untuk menentukan keakuratan, integritas, dan kualitas perangkat lunak komputer yang dikembangkan. Dalam pengujian perangkat lunak, perangkat lunak dijalankan dalam lingkungan eksperimental atau dunia nyata, menggunakan input yang dipilih dengan cara tertentu. Pada level rendah, pengujian perangkat lunak adalah aktivitas untuk memverifikasi bahwa sistem perangkat lunak Anda sehat. Pengujian perangkat lunak digunakan untuk memperbaiki kesalahan sehingga produk dirilis sebelum dirilis ke pengguna akhir. Sederhananya, pengujian

perangkat lunak pada dasarnya terdiri dari tiga bagian: persyaratan input, persyaratan hasil, dan bagian terkait. Pengujian perangkat lunak sangat penting. Ini untuk menghindari risiko kerusakan perangkat lunak.

a. Contoh kasus

Saat pesawat jatuh. Sebuah kecelakaan terjadi di China Airlines Penerbangan 140 pada 26 April 1994, menewaskan sebagian besar sejarah maskapai penerbangan China. Awalnya, pesawat dalam kondisi normal selama penerbangan, tetapi segera setelah lepas landas, menekan tombol lepas landas atau tombol putar sebelum mendarat di Nagoya menyebabkan pesawat gagal. Akhirnya, pesawat itu jatuh, mengklaim bahwa 264 orang tewas dan tujuh selamat dari kecelakaan tragis itu. Pengujian perangkat lunak yang sangat penting sedang dilakukan pada mesin akselerator medis Therac25. Pengujian dapat dikaitkan dengan interaksi antara antarmuka pengguna dan manusia dan komputer. Antarmuka pengguna yang buruk juga dapat menyebabkan hilangnya nyawa. Therac25 adalah perangkat terapi radiasi yang dirancang untuk menghancurkan pasien kanker. Mesin ini memiliki pemancar elektron dengan dua Konfigurasi.

1) Mode energi rendah, digunakan untuk memancarkan energi langsung ke pasiennya
2) Dalam mode energi tinggi pancarannya diblokir oleh filter pembangkit x-ray.

Permasalahannya adalah dalam perancangan sistem Therac25 terdapat race condition antara interface dan controller transmitter. Jika administrator hanya memilih satu mode dan perangkat memulai reset sistem, administrator akan memilih kembali mode lain setiap 8 detik-perangkat akan benar-benar menggunakannya untuk sumber magnet sistem Pindah ke posisi-menggunakan pengaturan baru yang sistem mungkin tidak mampu menerima. Oleh karena itu, bahkan seorang administrator berpengalaman dapat membunuh pasien jika dosisnya terlalu tinggi jika tidak diberikan secara hati-hati kepada pasien. Selain itu, pemrograman yang digunakan pada perangkat ini menemukan kesalahan fatal. Bug adalah kesalahan pemrograman dan perangkat tidak akan berfungsi dengan baik. Bug di perangkat ini dapat meningkatkan dosis radiasi hingga 10 kali lipat. Hal ini dapat menyebabkan keracunan pada pasien dan beberapa pasien dapat meninggal.

b. Prinsip pengujian

1) Ujicoba menunjukan adanya kecacatan (defect)
2) Ujicoba semua tidak akan di lakukan
3) Ujicoba paling pertama
4) Pengelompokan kecacatan *(Defect clustering)*
5) Paradoks Pestisida *(Pesticide paradox)*
6) Ujicoba yang bergantung pada situasi
7) Tak adanya kesalahan

c. Pengujian software

Biasanya terdiri dari verifikasi dinamis bahwa suatu program berperilaku dalam sekelompok kasus uji yang cocok, dipilih dengan benar dari rentang eksekusi yang tidak terbatas

1) Penguji dinamis menulis proses ke program input yang dipilih. Sistem yang kompleks dan non-deterministik berbeda dan bergantung pada status sistem, memungkinkan Anda untuk menentukan pengujian dengan nilai input yang akurat. Namun, dalam domain pengetahuan pengujian perangkat lunak, istilah "input" dapat dipertahankan, menyiratkan kesepakatan implisit bahwa artinya juga mencakup keadaan input yang ditentukan dalam situasi kritis. Rekayasa statis berbeda dengan pengujian dinamis. Teknologi statis terdaftar di area Pengetahuan Kualitas Perangkat Lunak. Perhatikan bahwa istilah komunitas yang berbeda tidak konsisten dan beberapa menggunakan istilah "tes" untuk merujuk pada teknologi statis.
2) Tes terbatas dilakukan pada subset dari semua kemungkinan tes yang ditentukan oleh kriteria risiko dan prioritas. Pengujian selalu bermakna antara sumber daya yang terbatas dan jadwal lawan.
3) Seleksi Banyak metode pengujian yang diusulkan berbeda secara signifikan dalam pemilihan rangkaian pengujian, dan insinyur perangkat lunak harus menyadari bahwa kriteria seleksi yang berbeda dapat menghasilkan tingkat efektivitas yang berbeda secara signifikan. Dan bagaimana menemukan jenis pilihan yang paling sesuai dengan kondisi Anda saat ini adalah masalah yang kompleks. Pada kenyataannya,

metode analisis risiko dan pengetahuan rekayasa perangkat lunak digunakan.

4) Diharapkan (diharapkan): Hal ini diperlukan untuk mengkonfirmasi apakah hasil tes program yang diamati dapat diterima, dan tidak selalu perlu untuk hanya menilai. Jika tidak, upaya pengujian tidak ada artinya. Perilaku yang diamati dapat ditunjukkan dalam hal persyaratan pengguna (biasanya disebut tes validasi), spesifikasi (tes validasi), atau perilaku yang diharapkan dari aturan atau harapan implisit (dijelaskan dalam domain pengetahuan)

Dalam beberapa tahun terakhir, perspektif pengujian perangkat lunak telah berhasil dan telah berkembang menjadi pendekatan yang konstruktif. Eksperimen yang menghilang setelah fase pengkodean untuk melacak bug selesai. Pengujian perangkat lunak, atau apa yang seharusnya, merangkum seluruh proses pengembangan dan pemeliharaan. Perencanaan pengujian perangkat lunak harus dimulai dengan proses persyaratan perangkat lunak. Perencanaan sistematis dari prosedur pengujian esensial dan rencana pengujian aktivasi desain untuk pengujian ini memberikan informasi yang berguna untuk desain perangkat lunak dan mengidentifikasi kelemahan potensial seperti perbedaan desain dan ketidakkonsistenan/ambiguitas dokumen. Saya dapat melakukannya. Tujuan dari pengujian perangkat lunak adalah:

1) Verifikasi dan validasi menentukan bug
2) Melacak kegagalan (fault)
3) Mendirikan rasa percaya dalam sebuah *software*
4) Memeriksa kembali atribut *software* (keandalan, kinerja, penggunaan memori, keamanan, kegunaan)

d. Fundamental Software Testing

Berbagai istilah dapat digunakan dalam literatur rekayasa perangkat lunak untuk memberikan contoh kesalahan. Yang paling penting dari kesalahan adalah kesalahan, kesalahan, kesalahan, dan sebagainya. Ada banyak nama untuk bug pada level yang berbeda. Kesalahan ditampilkan di tingkat programmer / pengembang. Error/bug dapat dilihat pada level test. Ini adalah kesalahan (yang dapat terjadi pada perangkat lunak atau perangkat keras) dan ditentukan pada tingkat pengguna / klien. Cacat adalah cacat

bawaan dan spesifikasi produk. Cacat ini mungkin karena kesalahan. Kesalahan adalah kesalahan dalam satu atau lebih baris kode. Kesalahan adalah keadaan perangkat lunak yang disebabkan oleh kesalahan. Kesalahan mungkin tampak tidak terprogram, menunjukkan bahwa perangkat lunak berfungsi seperti yang diharapkan pengembang. Baris program dapat dibiarkan tidak terpengaruh oleh tindakan, sehingga tidak terlihat seperti kesalahan, jadi mungkin cukup lama. Ini adalah kesalahan yang dapat terjadi ketika kesalahan terjadi. Ini adalah aktivitas manusia yang mengarah pada hasil yang salah. Karena kesalahan run-line, perangkat lunak melakukan apa yang tidak diinginkan pengembang, yang dapat menyebabkan jawaban yang salah. Kesalahan ini menyebabkan kesalahan (gagal). Kesalahannya adalah menyimpan hasil yang diharapkan dalam perangkat lunak. Selain itu, status kegagalan dapat didasarkan pada asumsi kegagalan yang ada. Kesalahan yang terjadi pada perangkat lunak adalah tidak berfungsinya sekumpulan perangkat lunak yang tidak menjalankan fungsinya.

1) Misalnya, kesalahan rilis perangkat lunak,
2) Perjalanan yang mengeksekusi dengan tidak normal,
3) Pengerjaannya dengan waktu dan tempat penyimpanan yang tidak normal,
4) dengan server yang penuh dan lain sebagainya.

Permasalahan penting dalam pengujian perangkat lunak, antara lain:

1) Kriterian seleksi tes / keriteria kecukupan tes *(Stopping Rules)*
   kriteria seleksi tes adalah alat yang digunakan untuk memilih kasus pengujian atau menentukan suatu rangkaian kasus pengujian yang cukup untuk tujuan yang ditentukan. Kridasi teria kecukupan pengujian bisa di pakai dalam menetapkan waktu pengujian yang memadai dapat dilakukan atau telah di selesaikan.
2) Validasi tes / tujuan tes
   Validasi uji dapat ditentukan dengan menganalisis implementasi beberapa program. Pilihan tes untuk dijalankan dapat didasarkan pada tujuan yang berbeda, dan efektivitas serangkaian tes hanya dapat dievaluasi berdasarkan tujuan yang dicapai.

3) Testing for Defect Discovery

jika pengujian berhasil, terjadi kegagalan sistem. Ini mungkin berbeda dari pengujian untuk menunjukkan bahwa perangkat lunak memenuhi spesifikasi atau atribut lain yang diperlukan. Ini karena tes berhasil jika tidak ada kesalahan yang diamati dalam kasus uji dan di lingkungan pengujian yang sebenarnya.

4) The Oracle Problem

Oracle adalah agen manusia atau teknisi yang mengambil tes untuk menentukan program yang benar dan dapat menyebabkan keputusan lulus atau gagal. Banyak jenis Oracle, termasuk spesifikasi persyaratan yang ambigu, model perilaku, dan komentar kode. Otomatisasi oracle mekanis biasanya sulit dan mahal.

5) Theoretical and Practical Limitations of Testing

Teori tes memperingatkan agar tidak memasukkan keyakinan yang tidak masuk akal ke dalam serangkaian tes. Sayangnya, hasil tes teoretis paling canggih adalah negatif karena menjelaskan tujuan yang tidak dapat dicapai dibandingkan dengan apa yang dicapai tes. Kutipan paling terkenal dalam konteks ini adalah moto Dijkstra: "Anda dapat menggunakan program uji untuk menunjukkan kesalahan, tetapi Anda tidak boleh melakukannya." Alasan yang jelas adalah bahwa pengujian terperinci tidak dapat dilakukan dengan perangkat lunak nyata. Oleh karena itu, pengujian harus berbasis risiko dan dapat dianggap sebagai strategi manajemen risiko.

6) The Problem of Infeasible Paths

Jalur aliran ontrol yang tidak dapat di eksekusi dengan data input apapun. ini adalah masalah yang penting dalam pengujian berbasis jalur, terutama dalam penurunan otomatis input pengujian untuk mengontrol jalur control aliran.

7) Testibilitas *(Testability)*

Jalur aliran kontrol yang tidak dapat dijalankan dengan data masukan. Ini adalah masalah penting dalam pengujian berbasis baca, terutama saat menurunkan input pengujian secara otomatis untuk mengontrol lead kontrol aliran.

e. Keterkaitan Penguji dengan aktivitas Lainya

Tes pada sebuah software dapat berkaitan dengan kegiatan dan sudut pandangan lainya, diantaranya:

1) Pengujian dan Teknik Kontrol Kualitas Perangkat Lunak Statis (terkait dengan Area Pengetahuan Kualitas Perangkat Lunak di bagian Teknik Kontrol Kualitas Perangkat Lunak).
2) Pengujian dan pembuktian keabsahan dan verifikasi formal (berdasarkan model dan metode di bidang rekayasa perangkat lunak).
3) Pengujian dan Debugging (berdasarkan Konstruksi Perangkat Lunak Area Pengetahuan di bagian Pengujian Konstruksi dan Landasan Komputasi Area Pengetahuan di bagian Alat dan Teknik Debug).
4) Program Pengujian dan Desain (Berdasarkan Area Pengetahuan Desain Perangkat Lunak di bagian "Tinjauan Desain").

f. Tujuan Pengujian software

Tujuan dari pengujian ini adalah untuk menemukan lebih banyak bug/bug pada produk tertentu. Tunjukkan produk perangkat lunak yang ditawarkan untuk memenuhi kebutuhan Anda. Perangkat lunak pemeriksaan kualitas verifikasi dengan biaya dan metode minimal. Untuk kasus pengujian berkualitas tinggi, lakukan pengujian yang efektif dan hasilkan laporan masalah yang tepat waktu dan berguna. Jenis-jenis pengujian perangkat lunak adalah:

1) Acceptance testing, tes penerimaan didasarkan pada persyaratan yang sudah di sepakati diawal. tujuan pada pengujian ini merupakan buat mendapat atau menolak aplikasi yg disediakan yang disediakan.
2) Instalation testing, tes instalasi umumnya sudah merampungkan tes system & tes penerimaan, pembuktian aplikasi selama proes instalasi pada lingkungan target. Pengujian instalasi bisa dipercaya menjadi pengujian system yg pada lakukan pada bawah batasan operasi misalnya lingkungan operasi konfigurasi perangkat keras & hambatan operasi konfigurasi perangkat keras, & proses instasi pula sudah diverifikasi.
3) α / β testing, beberapa organisasi pengguna pontensial menentukan buat menguji, pengujian alfa & pengujian beta tak jarang nir terawasi & nir termaksut didalam planning pengujian alpha (small group), seluruh ini pada lakukan sang tim penguji organisasi yg dikembangkan perangkat

lunak Beta (larger group), dilakukan sang sekelompok pengguna yg loyal terhadap perangkat lunak yg diujicobakan.

4) Reliability testing, Tes reability dilakukan menggunakan mengidentifikasikan & memperbaiki kesalahan. Selain itu, pengukuran statistik keandalan bisa diturunkan menggunakan membentuk perkara uji secara rambang dari profil operasi menurut metode terkini yg pada sebut pegujiaan operasional.
5) Regression testing adalah pengujian yg diulangi secara selektif terhadap suatu sistem atau komponen buat memverifikasi bahwa modifikasi akan mengakibatkan imbas yg nir diinginkan & bahwa sistem atau komponen masih memenuhi persyaratan yg ditentukan. Dalam praktiknya, metode ini buat menerangkan pada konteks pengujian bahwa aplikasi masih melewati pengujian yg sudah berlalu sebelumnya, yg terkadang dianggap pengujian non-regresi. Untuk pengembangan incremental, tujuan pengujian regresi merupakan buat menerangkan bahwa prilaku perangkat lunak nir akan berubah lantaran perubahan incremental pada aplikasi.

2. Teknologi Dalam pengujian White-box dan Black-box

Pengujian perangkat lunak adalah cara untuk menentukan apakah perangkat lunak yang Anda tulis berfungsi dengan baik. Tanpa pengujian pengujian perangkat lunak, Anda tidak dapat mengetahui apakah pengujian perangkat lunak memenuhi semua kriteria yang dibutuhkan pengguna Anda. Pengujian perangkat lunak ditentukan berdasarkan SDLC (Software Development Life Cycle) dan yakin akan mengambil alih eksekusi program, menjalankannya selama proses, dan menemukan kesalahan / kesalahan. Pengujian perangkat lunak ini merupakan tahap penting dan membutuhkan sumber daya yang memadai, yaitu sekitar 50% dari total biaya pengembangan perangkat lunak. Beberapa tujuan evaluasi perangkat lunak adalah untuk meningkatkan kualitas perangkat lunak, memeriksa persyaratan dan apakah sistem memenuhi persyaratan, dan melakukan identifikasi kekurangan/ *bug*, kesalahan, yaitu kesalahan yang terjadi pada perangkat lunak.

Dalam pengujian perangkat lunak, ada looh yang harus dilalui :

a. Analisis Permintaan
   Analisis tahap SDLC dari perangkat lunak yang lewat

b. Analisis Desain
   jenis Analisis ini merupakan bentuk dari analisis desain yang menganalisis komponen desain dan pra meter yang perlu diuji.

c. Test Desain
   Sempurnakan strategi pengujian yang akan dijalankan.

d. Run Test
   Menjalankan pengujian dan mencari kesalahan, bug dan kesalahan yang ada

e. Report Test
   Menyguhkan sebuah laporan dari hasil pengujian kepada pengembang dan memberikan kesimpulan apakah perangkat lunak tersebut layak untuk di gunakan.

a. White-box Testing

   Pengujian kotak putih adalah metode pemeriksaan modul untuk menguji aplikasi atau perangkat lunak untuk memeriksa dan menganalisis kesalahan kode program. Ketika modul dihasilkan dengan output yang tidak memenuhi syarat, kode dikompilasi ulang dan diperiksa hingga mencapai nilai yang diharapkan. Tes kotak putih meninjau dan menguji kode asli aplikasi / perangkat lunak, terlepas dari tampilan atau antarmuka pengguna aplikasi.

   1) Teknik White-box Testing
      a) Tes jalur dasar *(Basis Path Testing)*

         Pengujian jalur dasar adalah cara bagi perancang kasus uji untuk mengukur kompleksitas logis dari desain program dan menggunakan metrik ini sebagai panduan untuk menentukan serangkaian jalur eksekusi dasar. Kasus uji dibuat untuk menguji kumpulan inti untuk memastikan bahwa setiap pernyataan dalam program dapat dieksekusi setidaknya sekali selama pengujian.

      b) *Flow Graph*

         Flow grafik adalah symbol sederhana yang mewakili aliran control flow.

c) *Cyclomatic Complexity*

Ini menentukan jumlah semua jalur untuk mencari. Kompleksitas siklomatik adalah ukuran perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program. Perhitungan kompleksitas siklomatik menentukan jumlah jalur independen dalam program dasar dan meminimalkan jumlah tes yang perlu dilakukan untuk memastikan bahwa semua instruksi telah dieksekusi setidaknya sekali.

Cyclomatic complexity mempunyai dasar dalam teori graph dan bisa dihitung dengan tiga cara berikut :

(1) Region yang Jumlahnya sama dengan *cyclomatic complexity.*

(2) *Cyclomatic complexity*, V(G), untuk sebuah *flow graph*, G di definisikan sebagai : V(G) = E – N + 2, E adalah jumlah *edge* pada *flow graph*, dan N adalah jumlah node pada *flow graph*

(3) *Cyclomatic complexity*, V(G), untuk *flow graph*, G juga didefinisikan sebagai : V(G) = P + 1, P adalah jumlah predicate nodes yang terdapat pada *flow graph* G.

b. Graph Matrix

Berdasarkan mekanisme ini, proses digunakan untuk membuat diagram alir untuk menentukan set dasar jalur agar dapat diterima. Struktur data yang disebut matriks graf sangat berguna untuk mengembangkan perangkat lunak yang berguna untuk pengujian jalur dasar. Matriks graf adalah matriks kisi yang ukurannya (kardinalitas dan jumlah kolom) sama dengan jumlah simpul pada diagram alir. Di setiap baris dan kolom yang terkait dengan simpul yang diidentifikasi, data Matriks Data sesuai dengan koneksi (tepi) antara. Kerugian/Kelemahan White box-Testing:

1) Begitu mahal untuk dikerjakan karena membutuhkan penguji yang terampil untuk melakukan Tes.

2) Untuk software yang berukuran besar, metode *white box testing* ini di anggap boros karena membutuhkan resource yang banyak untuk di lakukan.

3) Tidak peduli dengan tampilan UI aplikasi.

c. Pengujian Kotak hitam *(Black-box Testing)*

Dalam pengujian *Black-Box Testing,* pengujian dilakukan berdasarkan informasi terperinci dari aplikasi, dan tampilannya seperti aplikasi. Fungsi yang terdapat pada aplikasi, dan alur fungsi sesuai dengan proses bisnis yang diinginkan oleh user. *Black-box testing* lebih kepada penguji ke tampilan luar (Interface) untuk kenyaanan pelangga. Tes ini tidak melihat dan menguji sumber program. Prinsip kerja *Black-box testing* adalah mengabaikan struktur control sehingga terfokus pada informasi domain.

1) Keuntungan dari *Black-box Testing* :
   a) User sebenarnya mengharuskan untuk memahami tentang Bahasa pemograman tertentu
   b) Digunakan untuk pwngujian dari suatu sudut pandang user agar dapat mengungkapkan inkosistensi atau ambiguitas dalam spesifikasinya.
   c) Pemprogram dan penguji saling ketergantungan.
2) Kekurangan Black-box Testing
   a) Jika tidak ada spesifikasi yang jelas, sulit untuk merancang pengujian kasus.
   b) Pengulangan tes yang sudah dilakukan kemungkinan dimiliki oleh programmer
   c) Beberapa bagian backend belum diuji sama sekali
3) Teknik Black-box Testing
   a) Partisi kesetaraan (Equivalence Partitioning)
      Prinsip kerja pada Teknik ini adalah dengan membuat partition atau membagi data input menjadi beberapa partisi,
   b) Analisi nilai batas(Boundary Value Analysis)
      Metode ini lebih menitikberatkan pada batasan, dimana terdapat sebuah kesalahan dari luar ataupun dari dalam software, minimum, atau maximum nilai dari kesalahan yang dapat di peroleh.
   c) Fuzzing
      Fuzz ialah merupakan Teknik untuk menemukan bug / gangguan dari perangkat lunak yang menggunakan injeksi data yang dianggap cacat ataupun sesi semi-otomatis
   d) Diagram sebab dan akibat (Cause-Effect Graph)

Cause-Effect Graph adalah suatu Teknik pengujiannya yang menggunakan graphic sebagai pacuannya. dalam diagram ini menggambarkan hubungan antara dampak dan penyebab dari kesalahan tersebut

e) Orthogonal Array Testing
Orthogonal Array Testing digunakan jika ukuran input domain yang relatif terbilang kecil, tetapi lumayan berat jika digunakan untuk skala besar dan anda dapat menggunakan pengujian arai orthogonal ini.

f) All Pair Testing
All Pair Testing pada Teknik ini, semua pasangan dari test case dirancang dalam bentuk yang berbagai rupa supaya bisa di eksekusi, yang memungkinkan sebuah kombinasi diskrit dari seluruh pasangan berdasar input parameternya. Yang bertujuan pengujian ini memiliki pasangan testcase yang meliputi semua pasangan tersebut.

g) State Transition
Pengujian ini berguna untuk melakukan ujicoba dalam keadaan mesin dan navigasi dari UI dalam bentuk grafik.

C. SOAL LATIHAN/TUGAS

1. Bagaimana proses pengujian pada uji coba validasi software ?
2. Sebutkan prinsip pengujian dalam uji coba valdasi software ?
3. Buatlah pengujian uji coba validasi software mengunakan metode white box ?
4. Buatlah ujicoba validasi software mengunakan metode black box ?
5. Apa peranan SDLC dalam uji coba validasi software ?

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

Therac25 adalah perangkat terapi radiasi yang dirancang untuk menghancurkan pasien kanker.

α / β testing, beberapa organisasi pengguna pontensial menentukan buat menguji, pengujian alfa & pengujian beta tak jarang nir terawasi & nir termaksut didalam planning pengujian alpha.

Cyclomatic complexity, V(G), untuk sebuah *flow graph*, G di definisikan sebagai : V(G) = E – N + 2, E adalah jumlah *edge* pada *flow graph*