PERTEMUAN 13
(VALIDASI SOFTWARE)

A. TUJUAN PEMBELAJARAN

Pada pertemuan kali ini akan mendeskripsikan tentang pemahaman Validasi Software. Kalian di haruskan mampu :

1. Menjelaskan Pengertian Validasi Software
2. Menjelaskan perencanaan Validasi Software
3. Menjelaskan Dasar-Dasar Ujicoba Validasi Software

B. URAIAN MATERI

1. Pengertian Validation Software

Validasi perangkat lunak *(software validation)* adalah bagian dari suatu validasi sistem yang terkomputerisasi *" computerized system validation "*(CSV), validasi sistem komputerisasi dapat didefinisikan sebagai bukti terdokumentasi dengan tingkat jaminan yang lebih tinggi bahwa perangkat lunak atau sistem terkomputerisasi, juga berfungsi sebagai desain sebuah perangkat lunak dan kebutuhan untuk penggunanya secara konsisten dan dapat di produksi.

Sistem terkomputerisasi juga dapat di artikan sebagai komputer yang mencangkup perangkat lunak, keras , dan periferal.yang di perlukan agar computer dapat berfungsi dengan baik.

Validation software (validasi perangkat lunak) sering juga di sebut verifikasi dan validasi. Verifikasi mengarah pada sekelompok aktivitas untuk memastikan bahwa program akan menunjukan suatu fungsi yang tepat dengan spesifikasinya. Validasi mengacu pada sebuah aktifitas kelompok berbeda yang memastikan bahwa software yang telah di kembangkan akan memenuhi suatu ekspetasi atau harapan pada si pemakai sistem. berpengaruh *review* dan *checking* pada sebuah proses dan ujicoba sistem. Uji coba pada sistem dapat mencangkup pergerakan program dengan skenario pengujian yang dapat di turunkan dari spesifikasi data yang nyata dan akan diproses oleh system.

Verification : " Are we building the product right ?"

Software seharusnya sesuai dengan spesifikasi.

Validation : " Are we building the right product ?"

Software seharusnya dapat melakukan apa yang benar-benar disyaratkan pada user

a. Proses Validasi dan Verifikasi

   Suatu Proses yang keseluruhanya melalui proses daur hidup V dan V, harus di terapkan untuk setiap tahapan melalui proses software yang akan dipunyai dua obyektif pada prinsip yaitu:

   1) kekurangan yang ditemui pada sebuah sistem,
   2) Memperkirakan apakah sistem berguna dan dapat di gunakan atau tidak dalam situasi oprasional.

b. Tujuan Validasi software

   1) Verifikasi dan validasi akan memberikan kepatian bahwa suatu software sesuai dengan tujuan.
   2) Hal ini bukan berarti benar-benar bebas dari kekurangan.
   3) Harus cukup baik untuk tujuan penggunaannya dan tipe dari penggunaan akan menentukan derajat kepastian yang di butuhkan.

c. Kapasitas Validasi dan Verifikasi

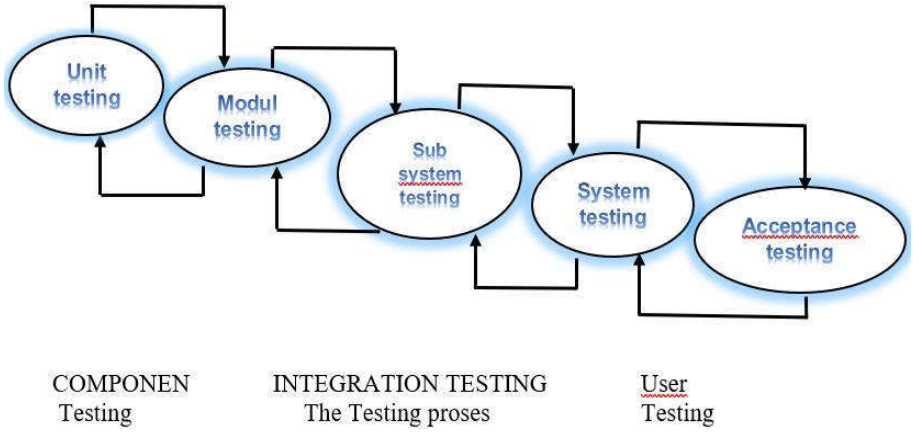
   Tergantung pada tujuan sistem, harapan user dan lingkungan pemasaran

   1) Fungsi Software Tingkat kepastian tergantung pada bagaimana kritikal softwareterhadap sebuah organisasi
   2) Harapan User mungkin mempunyai harapan yang rendang dengan software ynag ada
   3) Lingkungan Pemasaran Lebih awal melempar sebuah produk ke pasar lebih penting dari pada menemukan kekurangan dalam program

d. Proses Testing

   1) Component Testing Pengujian terhadap integrasi sub-system, yaitu keterhunbungan antara sub-sistem
   2) Acceptance Testing Pengujian terkahir sebelum sistem dapat di pakai oleh User Melibatkan pengujian dengan data dari pengguna sistem, Biasanya di kenal sebagai " Alpha Test " ("Beta Test " untuk software

komersial ,dimana pengujiannya dilakukan oleh pontensial customer").

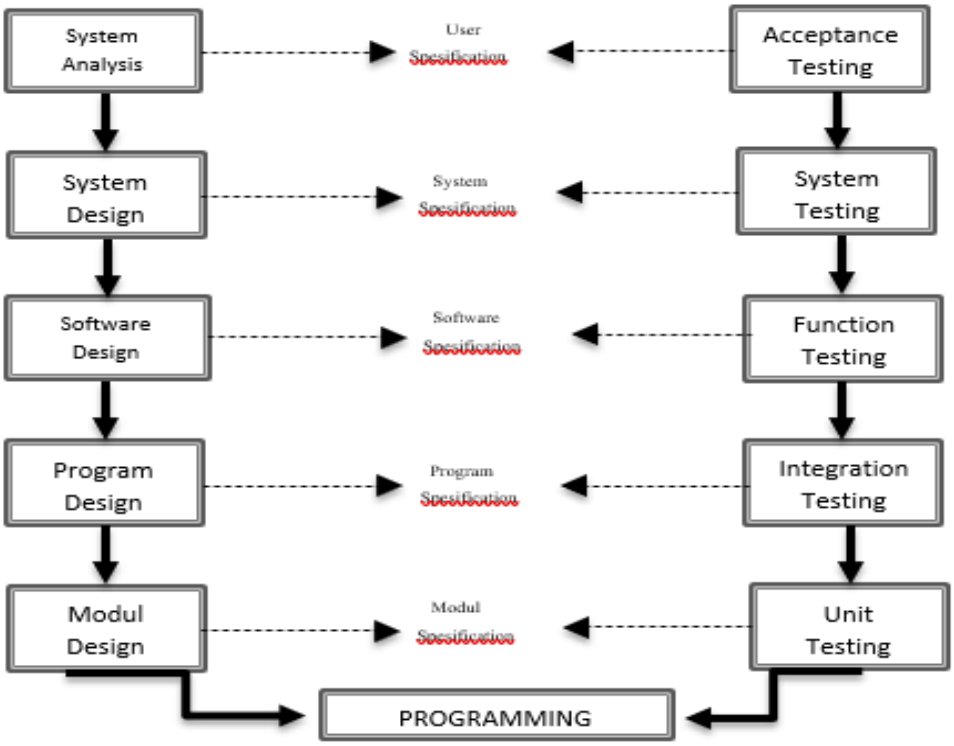


Gambar 13. 1 Proses Testing

a) Component testing adalah ujicoba komponen-komponen program yang biasanya dapat dilakukan oleh *component developer* (kecuali untuk sistem kritis).
b) Integration testing suatu pengujian pada beberapa kelompok komponen-komponen yang terintegrasi agar membentuk *sub-system* atau sistem. Yang akan di lakukan oleh team penguji yang independent. pengujian akan berdasarkan *specification system*.

e. Verifikasi Statik dan Dinamik

Inspeksi perangkat lunak. Berhubungan dengan Analisa representasi sistematik agar dapat ditemukannya masalah *(verifikasi static)*. dapat menjadi tambahan dari tool-based document dan code analysis, Software testing. Berhubungan dengan pelakasanaan dan memperhatikan prilaku produk (dinamic verifikasi). Sistem di jalankan dengan data tes prilaku oprasionalnya di perhatikan.



Gambar 13. 2 Pengembangan Model Verifikasi

2. Perencanaan Validasi

Rencana validasi dapat menentukan ruang lingkup dan bertujuan untuk project validasi. Rencana validasi ditulis pada awal project validasi atau terkadang dengan di tulis dengan spesifikasi persyaratan dan penggunanya. Biasanya untuk satu project validasi. Kumpulan dokumen yang di hasilkan selama project validasi dapat di sebut paket validasi. Setelah project validasi selesai, semua dokumen dalam paket validasi harus disimpan sesuai dengan prosedur kontrol dokumen pada situs yang anda sudah siapkan.

a. Contoh Rencana Validasi
   1) Hasil kerja atau dokumen yang akan di hasilkan selama proses validasi
   2) Sumber daya, departemen, dan personel untuk berpartisipasi dalam project validasi
      a) Jarak waktu untuk menyelesaikan project validasi
      b) Kriteria penerimaan agar dapat di pastikan agar sistem dapat memenuhi syarat yang di tentukan

c) Persyaratan ketentuan sistem, termaksud bagaimana sistem dapat memenuhi persyaratan ini

3) Rencana tersebut harus ditulis dengan sejumlah detail yang mencerminkan kompleksitas sistem.

4) Rencana tersebut harus di setujui, minimal, oleh pemiliksistem dan jaminan kualitas. Setelah disetujui, rencana tersebut harus dipertahankan sesuai dengan prosedur kontrol dokumen situs anda.

3. Dasar-dasar Ujicoba

a. Unit Testing/Module Testing

Pengujian Unit atau Modul Anda dapat menjalankan pengujian untuk menemukan logika modul atau kegagalan fungsional yang dapat diuji pada unit perangkat lunak terkecil, yaitu komponen yang saling berhubungan dan saling bergantung. Jenis yang ditampilkan dengan kesalahan termasuk:

1) Kesalahan Tampilan *(interface errors)*

a) Teting jenis ini dapat diaplikasikan pada suatu program dimana data control dapat berpindahkan dari modul satu menuju modul yang lainya.

b) Contoh yang sering di temukan menyertakan perpindahan suatu *control* dari modul ke subrutin / subprogram.

c) Objek dari suatu ujicoba adalah ditentukan bahwa suatu argument yang di lemparkan kesubrutin dapat sesuaikan pada parameter yang diterima.

d) Ujicoba ini dapat diaplikasikan untuk dipastikan kesesuaiannya jumlah field data dan atribut (ukuran dan tipe) field data dan suatu perintah pemindahan dan penerimaan.

2) Kesalahan Masukan / Keluaran *(input atau output errors)*

a) Ujicoba seperti ini dapat di aplikasikan agar dapat memastikan bahwa semua record (data eksternal akan di baca atau dituliskan) telah pindah dan di terima seperti yang di inginkan.

b) Ujicoba sebuah aplikasi pada atribut record,termakasud banyaknya field, ukuran field dan tipe data yang ada dalam field.

c) Untuk tambahan, kesalahan juga dapat terlihat pada format record, organisasi file, dan penggunaan kunci. Idenya adalah untuk memastikan kunci yang digunakan sesuai dengan record Dan file yang tersusun dan direfrensikan sesuai dengan kunci.

d) Ujicoba juga meliputi aturan pembukaan dan penutupan file yang digunakan dan penanganan kesalahan pada *input* dan *output.* Memeriksakan kesalahan pada *flagging* untuk kondisi *end-of-file* dan proses yang sesuai jika terjadi file *null* atau *empty.*

e) Untuk akses yang berlangsung pada file, ujicoba dapat di aplikasikan untuk permasalahan yang terjadi bila record yang sesuai dengan kunci di temukan , atau bukan.

f) fungsi ujicoba pada kesalahan input-output yang paling akhir adalah menemukan kesalahan pada output sistem, ialah prosedur ujicobanya untuk menghasilkan atau dapat menampilkan output yang sebenarnya.

g) Anda kemudian dapat memeriksa keluaran ini atau menyesuaikannya dengan spesifikasi modul untuk memastikan bahwa proses yang benar dilakukan dengan data dan hasil yang diharapkan dan bahwa keluaran sesuai dengan format keluaran.

3) Kesalahan Struktur Data *(data structure errors)*

a) Pengujian ini bertujuan untuk menemukan kesalahan dalam pengolahan dan pembentukan elemen data yang didefinisikan dalam modul proses. b) Contoh dapat disertai dengan tabel atau catatan sementara yang dapat digunakan untuk mentransfer data. Pengujian dalam aplikasi untuk format dan modifikasi atribut.

b) Contoh dapat disertai dengan tabel atau catatan sementara yang dapat digunakan untuk mentransfer data. Pengujian dalam aplikasi untuk format dan modifikasi atribut.

c) Tes lain mungkin terkait dengan definisi tabel, seperti langkah-langkah utama dan ukuran tabel.

d) Pemeriksaan integritas lainnya dilakukan. Artinya, ia menggunakan nama untuk mengurai penghitung dan ukuran untuk melengkapi spesifikasi elemen data.

4) Kesalahan Aritmatika *(Aritmatika errors)*
   a) Hasil dari intruksi perhitungan dalam modul ujicoba untuk menemukan suatu kesalahan mendefinisikan bagaimana mestinya seluruh data item yang termaksud dalam intruksi aritmatika.
   b) Pada data dan item, pengujian dapat diaplikasikan untuk di pastikannya data berada didalam keadaan yang seharusnya untuk di eksekusi intruksi, dengan ukuran dari hasil sementara dan hasil akhir mungkin cukup besar untuk mengakomodasikan dari hasil perhitungan (eliminasi kesalahan pontensial yang di sebabkan pemotongan), dalam perhitungan dapat dilaksanakan dengan perintah yang pasti untuk hasil yang telah di spesifikasikan sebelumnya, dan bahwa untuk nilai nol tidak dapat digunakan sebagai pembagian. (bila ditemukan dan dibagi dengan nol, maka program harus bisa menangani kondisi seperti seharusnya).

5) Keasalahan perbandingan *(Comparison errors)*

   a) Pengujian ini dapat mencari kesalahan yang meliputi presentasi data item yang beda tipe atau suatu bentuk untuk fungsifungsi pembandingan.
   b) Bertujuan agar dapat dipastikan bahwa program atau modul tidak akan bisa mengizinkan suatu operasi perbandingan antara field alphabetik dengan field numrik atau antara field numerik dengan bentuk yang berbeda,
   c) Sama seperti fungsi pengujian kesalahan lainya, idenya yaitu untuk menyajikan kondisikondisi tersebut kesistem dan memastikan bahwa sistem dapat di tangani dan merecovernya.
   d) Pengujian akan menjadi lebih sulit untuk menetapkan fungsi seperti operasi perbandingan jamak bersarang *(multiple nested comparison).*

6) Prinsip yang di gunakan pada ujicoba ke modul ialah:

   a) Ujicoba semua memungkinan kombinasi untuk eksekusi atau jalur logika
   b) Ujicoba pada semua titik masuknya dan keluarnya untuk setiap modul.
   c) Ujicoba semua perintah sedikitnya satu kali

d) Ujicoba keseluruh pengumpulan yang meliputi seluruh range pada indeks atau subscript-nya

b. Sub Sistem Pengujian *(Integration Testing)*

Integrasi program mencakup metode yang disertakan untuk menghubungkan modul untuk membentuk subsistem atau sistem yang lengkap. Pengujian integrasi didasarkan pada spesifikasi sistem dan menguji hubungan antar program untuk menentukan potensi dan kekuatan implementasi. Saat menguji antarmuka antar modul dalam program aplikasi Anda, Anda dapat menggunakan pengujian untuk memprioritaskannya. Ada dua pendekatan yang bisa diterapkan:

1) Uji inkremental. Ini adalah modul yang dapat Anda tambahkan ke modul lain untuk pengujian individual, biasanya seperti modul baru.
2) Uji non-incremental. Semua modul dalam sebuah program dapat dibuat terlebih dahulu kemudian digabungkan dan diuji sebagai satu unit, sehingga semua antarmuka dalam modul pengujian diuji terhadap semua program pada saat yang bersamaan.

Ada dua metode yang dapat digunakan dalam uji inkremental, yaitu :

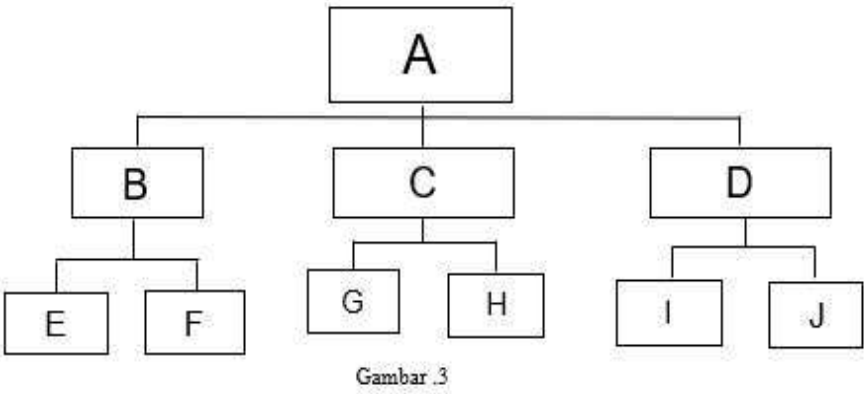
a) Top-down testing

(1) Berlaku untuk modul berdasarkan basis top-down atau kontrol hierarki ke bawah, atau untuk proses dari awal hingga detail modul modul tingkat atas.
(2) Komponen tingkat atas dari sistem terintegrasi, diuji sebelum desain dan implementasi penuh.
(3) Komponen tingkat atas dari sistem terintegrasi, diuji sebelum desain dan implementasi penuh.
(4) Integrasi dengan Depth First. Ini mengintegrasikan modul ke jalur utama. B. Dari Gambar A di bawah,
(5) B dan E pertama terintegrasi, kemudian terhubung ke jalur kiri dan kanan
(6) Integrasi Ibreadthfrist langsung mengintegrasikan semua modul dependen. Misalnya, A, B, dan C (jahitan D) digabung dulu, lalu ke level di bawah
(7) Proses integrasi di lakukan pada 5 tahapan

(a) Selanjutnya, modul konfigurasi utama digunakan sebagai test driver dan stub dari semua modul, yang merupakan modul kontrol regulasi langsung, diganti.
(b) Pendekatan integrasi yang dipilih menggantikan rintisan dependen dengan modul aktual pada saat yang sama.
(c) Pengujian dijalankan pada modul per integrasi.
(d) Pada akhir setiap rangkaian pengujian, stub lainnya diganti dengan modul sebenarnya.
(e) Anda dapat menjalankan uji regresi (menjalankan beberapa atau semua eksperimen sebelumnya) untuk menentukan apakah kesalahan baru telah terjadi.

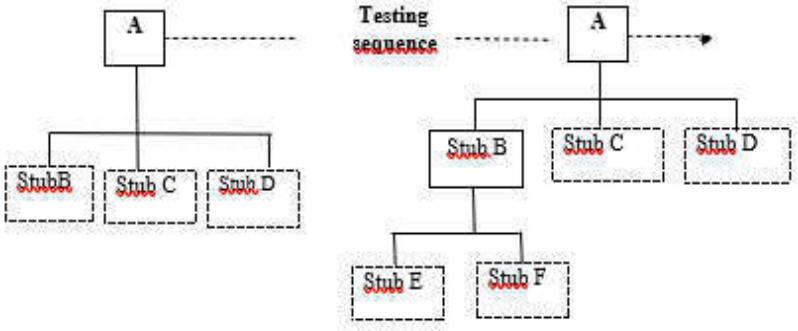
(8) Proses berulang dari langkah kedua melalui seluruh struktur program hingga pembentukan program.
(9) Hubungan dari modul pertama ke modul anak yang tidak diimplementasikan. Oleh karena itu, beberapa modul yang menangani antarmuka dikerahkan dan digunakan dalam menjalankan tes yang diterapkan untuk terhubung ke modul induk melalui modul anak.
(10) Pengujian ini sering disebut stub. Mensimulasikan proses yang berjalan ketika modul dikodekan.
(11) Selama pengujian, target stub dapat digunakan untuk memanggil subrutin sehingga modul tingkat yang lebih tinggi dapat melakukan fungsi kontrol. Rintisan dapat meneruskan data ke manajer untuk menguji antarmuka.
(12) Pengujian pendekatan top-down menyediakan validasi awal antarmuka utama dan semua logika kontrol yang perlu diperoleh dari tingkat atas struktur program.
(13) Mulai dari atas ke bawah memungkinkan Anda untuk mendemonstrasikan fungsionalitas program sepenuhnya dan sepenuhnya pada titik awal dan urutan pengujian.

Perhatikan gambar berikut:



Gambar .3

Bagan struktur program yang akan diuji secara bertahap



Gambar 13. 3 Proses Testing

b) Bottom-up

(1) Percontohan dimulai pada tingkat terendah dari struktur program dan naik sebagai rangkaian modul yang terintegrasi dan teruji.

(2) Prinsip-prinsip yang dapat mencegah top-down yang sama juga diterapkan dan diuji. Bottonup dimulai di area entri dan berakhir sebagai strategi pengujian keseluruhan.

(3) Penyelam diuji pada level terendah. Pengemudi sama dengan rintisan, tetapi sebaliknya. Driver adalah modul yang sering diuji untuk memanggil submodul dan mengirim data melalui

antarmuka. Pengemudi mensimulasikan pergerakan kelompok yang belum naik dari koordinat teratas modul yang akan diuji.

(4) Keuntungan dari pendekatan bottom-up adalah untuk pra-kontrol dan mengidentifikasi aliran proses rinci yang dapat terjadi antara antarmuka tingkat rendah dan kasus masukan dan keluaran pelatihan.

(5) Kelemahannya adalah pendekatan menunda tampilan seluruh rangkaian program sampai semua pengujian modul telah ditentukan. Anda dapat menerapkan strategi integrasi bottom-up dengan mengikuti langkah-langkah berikut:

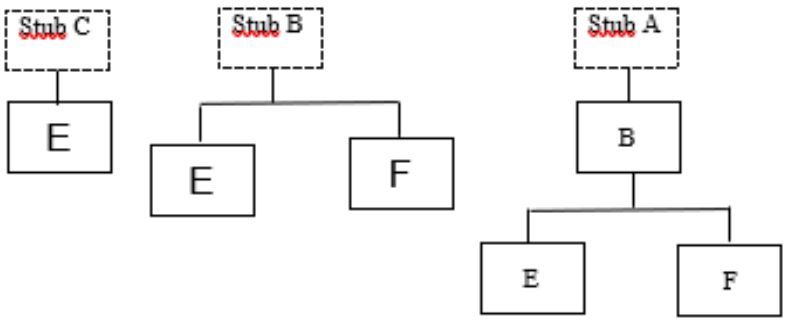
   (a) Modul tingkat rendah dapat digabungkan menjadi sebuah cluster atau disebut build yang dapat menampilkan sub-fitur perangkat lunak khusus.

   (b) Driver (program kontrol uji) dibuat untuk menyesuaikan kasus uji input dan output.

   (c) Cluster akan diuji

   (d) Driver akan dihapus dan cluster akan dikelompokkan ke atas sesuai dengan struktur program.

Perhatikan gambar berikut:



Gambar 13. 4 Pola pengujian incremental bottom-up

Ada 4 hal penting yang membedakan top-down dengan bottom up :

a) *Architecture validation*, pengujian top-down mencari kelemahan dalam arsitektur sistem, dan tingkat atas dirancang pada awal proses pembentukan. Ini biasanya berarti bahwa semakin cepat cacat struktural dikenali, semakin rendah biayanya. Pengujian desain top-down bottom-up, di sisi lain, hanya divalidasi selama fase akhir proses.

b) *System demonstration*, sistem dan pengujian top-down yang menunjukkan sistem yang sudah berjalan. Sementara itu, jika sistem Anda terdiri dari komponen daur ulang, Anda dapat menjalankan pengujian melalui demo sistem dari bawah ke atas.

c) *Test implementation,* Pengujian top-down sulit karena diimplementasikan karena rintisan program tingkat rendah dari sistem yang perlu Anda buat saat menggunakan pengujian bottom-up. Pengemudi tes perlu menulis ke tingkat rendah untuk berlatih. Dengan menguji driver ini, Anda dapat mensimulasikan komponen lingkungan dan memanggil komponen yang ingin Anda uji.

d) *Test Observation*, baik uji coba top-down maupun bottom-up, dapat menimbulkan masalah dengan uji coba observasi. Dari atas ke bawah, dari tingkat atas, sistem yang diterapkan pertama kali tidak dapat menghasilkan output. Untuk menguji level ini, dibuat untuk dapat menghasilkan output. Demikian pula sebaliknya berlaku untuk pendekatan bottom-up.

1) Combined Testing Methods

a) Tidak ada aturan baku kapan harusmelakukan pengujian dengan pendekatan *top-down* atau *bottom-up* dilasanakan. Itu juga dapat di gunakan dalam kombinasi dengan keduanya.

b) Semua pada dasarnya pendekatan modul-ke-modul. penting untuk menemukan rencana untuk menguji integrasi incremental dari hubungan dan antarmuka yang ada di semua modul di sistem.

2) Fungsi Testing

Pengujian fungsional untuk menemukan kesalahan atau kegagalan paket perangkat lunak lengkap menggunakan teknologi kotak hitam. Pada

dasarnya, pengujian fungsional mengikuti pengujian modul dan pengujian integrasi. Anda tidak dapat menguji sistem Anda sampai semua modul telah diuji dan disetel. Dalam hal ini, modul juga perlu diintegrasikan secara individu atau kelompok. Aturan khusus pengujian fungsional adalah melatih dan memantau seluruh program untuk memastikan bahwa spesifikasi input dan output pengguna eksternal terpenuhi..

Jalankan fungsi program untuk memverifikasi bahwa perangkat lunak berjalan sebagaimana dimaksud. Catatan input digunakan untuk pengujian fungsional yang mencakup bidang data benar dan salah. Ini termasuk kasus di mana nilai data alfanumerik lebih panjang dari bidang yang ditentukan, atau di mana penempatan nilai yang salah, seperti nilai data numerik, ditetapkan ke bidang alfanumerik. Anda dapat memeriksa parameter input dan output untuk melihat nilai elemen data benar atau salah dan menentukan kemampuan program Anda. Untuk proses.

3) Kriteria ujicoba validasi

   a) Validasi perangkat lunak didapatkan melewati serangkaian pengujian black box yang didemonstrasikan sesuaian dengan permintaan atau kebutuhan.
   b) Melewati setiap pengujian fungsi atau validasi dilakukan ada beberapa kondisi yang memungkikan terjadinya : (1.) karakteristik dengan fungsi atau peforma sesuai dengan spesifikasinya dan diterima atau (2.) menemukan penyimpangan dari spesifikasi dan menyebabkan cacat.

4) Review Konfigurasi

Bagian penting dari proses validasi adalah tinjauan konfigurasi. Tujuan tinjauan konfigurasi ini adalah untuk membuat katalog semua konfigurasi perangkat lunak yang sudah mapan dan menyertakan detail penting untuk mendukung fase pemeliharaan siklus hidup perangkat lunak.

C. SOAL LATIHAN/TUGAS

1. Jelaskan difinisi dari Validasi perangkat lunak ?
2. Sebutkan proses Component testing ?
3. Sebutkan Integrasi program mencakup metode validasi proses ?

4. Ada berapa spesifikasi persyaratan dalam validasi software ?
5. Bagaimana Sub Sistem Pengujian pada valsidasi 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

computerized system validation (CSV), validasi sistem komputerisasi dapat didefinisikan sebagai bukti terdokumentasi dengan tingkat jaminan yang lebih tinggi bahwa perangkat lunak atau sistem terkomputerisasi

.*Architecture validation*, pengujian top-down mencari kelemahan dalam arsitektur sistem, dan tingkat atas dirancang pada awal proses pembentukan.

Verifikasi Statik dan Dinamik Inspeksi perangkat lunak. Berhubungan dengan Analisa representasi sistematik agar dapat ditemukannya masalah *(verifikasi static)*.