MODUL PENGAJARAN TESTINGDAN IMPLEMENTASI

Penulis
Dani Rukmana Manday
Mardi Turnip
Stiven Hamonangan Sinurat

MODUL PENGAJARAN TESTING DAN IMPLEMENTASI

Penulis:

Dani Rukmana Manday

Mardi Turnip

Stiven Hamonangan Sinurat

Editor:

Jepri Banjarnahor

ISBN:

Penerbit
UNPRI PRESS

Jl. Sampul, Medan

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat yang telah diberikan, sehingga penulisan buku monograf ini dapat diselesaikan.

Buku ini membahas tentang testing implementasi. Pemahaman yang mendalam tentang testing implementasi sangat penting dalam dunia teknologi informasi. Buku ini secara detail menjelaskan peran teknologi testing implementasi dalam menghubungkan dan mengelola perangkat komputer.

Buku ini diharapkan dapat meningkatkan pemahaman pembaca tentang pengaruh teknologi testing implementasi terhadap kinerja perangkat komputer dan komunikasi antara perangkat tersebut. Ucapan terima kasih penulis sampaikan kepada semua pihak yang telah mendukung penerbitan buku ini.

Penulis menyadari bahwa buku monograf ini masih jauh dari kata sempurna, dan dengan kerendahan hati, penulis siap menerima kritik dan saran yang membangun terkait buku monograf ini. Semoga buku ini dapat memberikan kontribusi positif dalam pemahaman dunia testing implementasi. Akhir kata, penulis mengucapkan terima kasih banyak kepada semua pihak yang terlibat dalam penyelesaian modul ini.

Medan, Maret 2023

Penulis

Daftar Isi

DAFTAR GAMBAR

TESTING & IMPLEMENTASI SISTEM

Pengantar

Testing dan implementasi system informasi

a. Melakukan pengujian terhadap system informasi / program aplikasi/aplikasi sebelum digunakan.
b. Menguji dan membandingan dengan system sebelumnya, untuk memunculkan keunggulan pada system yang lama dan mengurangi kesalahan pada system yang baru. System yang baru lebih baik dari pada system yang lama.
c. Mervisi system yang diuji, sampai system benar benar dapat menyelesaikan maslah pada system / organisasi (revisi system sebelum system digunakan).
d. Sistem yang sudah digunakan, berarti sudah melalui pengujian system dan system layak dioperasikan / digunakan.

Defenisi Pengujian Sistem

a. Suatu proses yang dilakukan untuk menilai apakah yang dirancang telah sesuai dengan apa yang diharapkan.
b. Suatu kegian untuk mengevaluasi keunggulan dan kelermahan terhadap sesuatu yang diuji (kwalitas produk).
c. Mengevaluasi terhadap urutan kegiatan yang sistematis dalam mencapai tujuan system.
d. Mengevaluasi keseimbangan jumlah pelaksanaan kegiatan dengan beban kerja dalam sesuatu prosedur kegiatan.

Pengujian Sistem

a. Melakukan proses evaluasi terhadap system yang sudah ada apakah system sudah sesuai yang dharapkan user.
b. Menilai dan mengevaluasi terhadap output atau ahasil system.
c. Menguji terhadpa input, pengelolaan (proses)dan output system.
d. Melakukan penilaian dan evaluasi terhadpat komponen system prosedur pelaksanaan kegiatan dan mutu atau kwalitas hasil system.

Pengujian terhadap sistem

1. **Personil**
   a. Personil ditempakan sudah sesuai dengan skill atau kemampuan yang dimilikinya.
   b. Beban kerja yang optimum untuk masing masing personil.
   c. Loyalitas atau kemamuan bekerja sama untuk menyelesaikan suatu kegiatan.
   d. Kemampuan personil dalam menyelesaikan masalah.
2. **Pengujian kegiatan**
   a. Prosedur dan system kerja yang sistematis.
   b. Perencanaan yang terkontrol dan terjadwal.
   c. Arah tujuan atau target ang dapat dilaksanakan sesuai dengan perencanaan.
   d. Hasil kegiatan yang terukur.
   e. Kesemimbangan kegiatan dengan bersarnya biaya yang digunakan.
3. **Pengujian misi atau tujuan**
   a. Adanya integrasi antara personil yang terlibat dengan kegiatan yang dilaksanakn dalam mencapai target system
   b. Kwaliatas dari kegiatan yang mewujudkan tujuan system.

Tujuan Testing dan Implementasi

1. Tujuan Testing dan Implementasi

   Melakukan pengujian terhadap system informasi apakah sudah memenuhi kebutuhan user atau system informasi sudah layak digunakan dengan melalui :

   1) Uji analisis
   2) Uji perancangan
   3) Uji implementasi

2. Sasaran

   Aplikasi system informasi yang sudah melakukan uji kelayakan system, maka aplikasi system informasi akan terjadi peningkatan :

   1) Performance / kinerja
   2) Information / nilai mutu informasi
   3) Economi / nilai ekonomis
   4) Control / pengendalian diri
   5) Eficeinsi
   6) Service / pelayanan

Testing dan Sistem Informasi

Elemen kritis dari jaminan kwlitas dan mempresentasikan kajian pokok dari spesifkasi, desain dan pengkodean atau pngujian terhadap aplikasi system informasi berdasrkan komonen. System informasi yaitu pengujian model, pengujian output, database, teknologi dan pengujian kendali terhadap aplikasi system informasi. Sehingga memberikan system informasi yang berkwalitas terhadap user. Testing system informasi atau pengujian system informasi à pengujian terhadap aplikasi system informasi (software application) Program aplikasi.

a. Pengujian sintax
b. Pengujian lgika dan proses

c. Pengujian output Yaitu program bebas dari kesalahan, keraguan dan kegagalan.

Pengujian terhadap komponen system informasi

1. Model / proses Menguji hasil rancangan tentang proses atau moder dari suatu system informasi yang akan dbuat dengan program aplikasi. Contoh menguji flow chart KRS.
2. Output Menguji hasil rancangan dengan laporan yang disajikan aplikasi, apakah sudah memberikan informasi sesuai dengan kebutuhan user. Contoh KRS (MHS,DOSEN, PA,MTK,DOSEN PA, JADWAL, IPK, dll).
3. Input Pengujian terhadap jumlah atau variable data yang dibutuhkan output, contoh data MHS, data DOSEN, MTK, dll.
4. Database Menguji atribtu data , relasi antar file atau hubungan elemen data, julah database yang sesuai untuk menghaslkan output dan menampung data sesuai dengan form entri.
5. Teknologi Pengujian terhadap kemampuan peralatan yang digunakan, kemapuan software dalam menjalankan aplikasi, sehingga mampu memberikan informasi yang iteraktiv kepada user.contoh SS, SO, SA.
6. Control atau kendali Menguji terhadap keamanan dan hak akses dari aplikasi dan dapat memonitor.

Kesimpulan

Software testing adalah aktivitas-aktivitas yang bertujuan untuk mengevaluasi atribut-atribut atau kemampuan sebuah program atau sistem dan penentuan apakah sesuai dengan hasil yang diharapkan. Testing adalah proses pemeriksaan program dengan tujuan tertentu dalam menemukan kesalahan sebelum diserahkan ke pengguna.

Tahapan Testing : Terdapat cukup banyak pendekatan yang

dilakukan untuk melakukan testing. Salah satu definisi testing adalah "sebuah proses yang melakukan pertanyaan terhadap sebuah produk untuk dinilai", di mana "pertanyaan" merupakan segala sesuatu yang diberikan kepada produk sebagai pengujian. Beberapa tahapan testing yang umum dilalui oleh aplikasi adalah sebagai berikut:

1. Unit/ComponentTesting. Terbagi atas testing terhadap unit dan component. Unit testing merupakan proses testing, di mana Anda melakukan testing pada bagian basic dari kode program. Contohnya adalah memeriksa kode program pada event, procedure, dan function. Unit Testing meyakinkan bahwa masing-masing unit tersebut berjalan sebagaimana mestinya.Pada Unit Testing, Anda memeriksa bagian kode program secara terpisah dari bagian yang lain. Anda dapat langsung melakukan Unit Testing setiap kali sebuah kode unit (event, procedure, function) selesai dibuat. Anda dapat memeriksa kode unit dengan menjalankannya baris per baris untuk memastikan bahwa proses yang dilakukan berjalan sebagaimana yangAndainginkan.
2. IntegrationTesting. Setelah Anda melakukan Unit/Component Testing, langkah berikutnya adalah memeriksa bagaimana unit-unit tersebut bekerja sebagai suatu kombinasi, bukan lagi sebagai suatu unit yang individual. Sebagai contoh, Anda memiliki sebuah proses yang dikerjakan oleh dua function, di mana satu function menggunakan hasil output dari function yang lainnya. Kedua function ini telah berjalan dengan baik secara individu pada Unit Testing. Pada tahap Integration Testing, Anda memeriksa hasil dari interaksi kedua function tersebut, apakah bekerja sesuai dengan hasil yang diharapkan. Anda juga harus memastikan bahwa seluruh kondisi yang mungkin terjadi dari hasil interaksi antarunit tersebut

menghasilkanoutputyangdiharapkan.

3. SystemTesting. Mencakup testing aplikasi yang telah selesai didevelop. Karena itu, aplikasi harus terlihat dan berfungsi sebagaimana mestinya terhadap end-user atau pengguna akhir. Untuk itu, testing dilakukan dengan menggunakan data yang menggambarkan data yang digunakan oleh pengguna sesungguhnya terhadap aplikasi. Jika aplikasi Anda di-develop untuk lingkungan yang besar, Anda dapat melakukan testing pada dua komputer yang berbeda. Komputer yang Anda gunakan sebagai komputer testing harus terlebih dahulu dikonfigurasi hanya dengan: a. Operating system yang dibutuhkan. b. Driver yang diperlukan oleh aplikasi. c. Aplikasi yang dites. Dengan menggunakan konfi gurasi yang paling minimal dan sederhana, maka dapat membantu nda untuk memastikan bahwa permasalahan yang timbul selama testing berlangsung adalah merupakan kesalahan aplikasi, dan bukan kesalahan yang berasal dari aplikasi atau software lain.

4. AcceptanceTesting. Seperti Integration Testing, Acceptance Testing juga meliputi testing keseluruhan aplikasi. Perbedaannya terletak pada siapa yang melakukan testing. Pada tahap ini, end-user yang terpilih melakukan testing terhadap fungsi-fungsi aplikasi dan melaporkan permasalahan yang ditemukan. Testing yang dilakukan merupakan simulasi penggunaan nyata dari aplikasi pada lingkungan yang sebenarnya. Proses ini merupakan salah satu tahap final sebelum pengguna menyetujui dan menerima penerapan sistem aplikasi yang baru. Karena itu pada tahap ini sudah tidak difokuskan untuk mengangkat permasalahan kecil seperti kesalahan pengetikan, ataupun kosmetik aplikasi. Hal-hal minor seperti di atas sudah seharusnya ditangani selama Unit/Component Testing dan Integration Testing.

5. RegressionTesting. Merupakan bagian penting dari masing-masing tahap proses testing. Regression Testing mencakup pengujian ulang terhadap unit, component, proses, atau keseluruhan aplikasi setelah perbaikan suatu kesalahan dilakukan.Regression Testing memastikan permasalahan yang terjadi telah ditanggulangi, dan tidak terdapat permasalahan baru yang timbul sebagai efek perbaikan tersebut. Selain itu, tahap ini tidak hanya berguna untuk melakukan pengujian aplikasi, tetapi dapat juga digunakan untuk melakukan pemantauan kualitas dari output yang dihasilkan. Sebagai contoh, Regression Testing memantau ukuran file, waktu yang dibutuhkan untuk melakukan suatu tes, waktu yangdibutuhkanuntukmelakukankompilasi,danlainsebagainya.

PROSES REKAYASA PERANGKAT LUNAK

Pengertian SDLC (Software Development Life Cycle) LUNAK

Pengertian SDLC (Software Development Life Cycle) SDLC (Software Development Life Cycle) adalah kerangka kerja atau model manajemen proyek terstruktur yang menguraikan fase-fase yang diperlukan untuk membangun sistem TI, dari awal hingga hasil akhir. Tujuan dari Software Development Life Cycle adalah untuk menciptakan proses produksi yang efektif dan berkualitas tinggi agar dapat memenuhi atau melampaui harapan klien sesuai dengan anggaran dan jadwal yang telah ditentukan. Kerangka kerja ini sudah banyak digunakan oleh berbagai perusahaan IT baik itu perusahaan besar ataupun kecil. Dengan tetap patuh terhadap kerangka kerja SDLC ini, maka perusahaan dapat mempercepat proses pengembangan dan meminimalkan risiko proyek terkait waktu dan biaya yang diperlukan.

Mengapa SDLC Penting? Apa Keuntungan Menerapkan SDLC?

a. SDLC dapat membantu perencanaan, estimasi, dan penjadwalan proyek dengan baik

b. Kontrol proyek dapat dilakukan dengan lebih mudah

c. Semua pemangku kepentingan dapat mengetahui bagaimana siklus pengembangan software secara transparan

d. Dapat mempercepat proses pengembangan software
e. Mengurangi risiko proyek
f. Dapat mengurangi biaya manajemen proyek dan biaya produksi g. Meningkatkan hubungan yang baik dengan klien

Tahapan SDLC (Software DevelopmentLife Cycle)

Setelah Anda memahami apa pengertian SDLC serta tujuannya, tentu Anda ingin mengetahui bagaimana siklus atau tahapan yang diperlukan ketika menjalankan kerangka kerja ini. Secara garis besar terdapat beberapa tahapan SDLC yang perlu Anda lakukan yaitu:

a. Planning

Planning adalah tahap perencanaan dimana tim akan mengidentifikasi dan menentukan scope atau ruang lingkup yang perlu dilakukan dalam proses pengembangan proyek. Pada tahap ini, tim juga akan mengumpulkan semua informasi yang dibutuhkan dalam proses pengembangan software dari para pemangku kepentingan. Setelah itu, tim akan merencanakan struktur tim, time frame, budget, security, dan berbagai faktor penting lain yang dibutuhkan untuk pengembangan software.

b. Analysis

Tahapan SDLC yang selanjutnya adalah proses analisis. Pada tahap ini, tim akan menganalisis kebutuhan fungsional sistem. Jadi, tim akan melakukan analisis untuk mengetahui apa masalah bisnis, apa target yang ingin dicapai, apa tujuan utama dari pengembangan software tersebut, apa fungsi dari software yang akan dikembangkan, dan lain-lain. Analisi ini diperlukan dalam tahapan SDLC agar produk nantinya akan memiliki hasil akhir yang sesuai

c. Design

Berdasarkan requirement yang telah ditentukan sebelumnya, maka tim akan membuat rencana desain atau spesifikasi desain. Beberapa aspek desain yang akan ditentukan seperti:

- Architecture: bahasa pemrograman yang akan digunakan, desain software secara keseluruhan, dan lain-lain.
- User Interface: mendefinisikan bagaimana cara users ketika berinteraksi dengan software serta bagaimana cara software memberikan respon.
- Platform: platform tempat software dapat berjalan seperti Android, iOs, Linux, dan lainlain.
- Security: langkah-langkah untuk mengamankan sistem software seperti enkripsi lalu lintas SSL, perlindungan kata sandi, atau yang lain.

Rincian desain tersebut kemudian akan dibahas dengan para pemangku kepentingan. Tim akan menjelaskankan dengan berbagai parameter seperti risiko, teknologi yang akan digunakan, kapabilitas tim, kendala proyek, waktu dan anggaran. Setelah itu, pemangku kepentingan akan meninjau kembali desain tersebut dan menawarkan umpan balik dan saran.

d. Development

Dalam fase ini, proses pengembangan software dimulai. Jadi, tim pengembang akan mulai membangun seluruh sistem dengan menulis kode menggunakan bahasa pemrograman yang dipilih. Tahapan SDLC ini dapat dikatakan sebagai fase terpanjang dari proses pengembangan software. Untuk pengerjaan proyek besar, proses pengembangan software biasanya akan dibagi menjadi beberapa unit atau modul kemudian ditugaskan ke beberapa tim pengembang. Database admin akan membuat data yang diperlukan dalam database, front-end developer bertugas membuat UI dan GUI untuk berinteraksi dengan back-end. Proses pengembangan

software tersebut akan dilakukan berdasarkan pedoman dan prosedur yang sudah ditentukan sebelumnya.

e. Testing

Tahapan SDLC ini akan melibatkan para software Quality Assurance (QA) untuk melakukan pengujian pada sistem dan menilai apakah software dapat bekerja sesuai dengan fungsionalitas yang diharapkan. Tim QA akan menguji semua area software untuk memastikan bahwa sistem terbebas dari cacat, error, ataupun bug. Jika ternyata masalah ditemukan di dalam software yang dikembangkan, maka tim QA akan menginformasikannya dengan tim pengembang agar perbaikan dapat segera dilakukan. Proses ini berlanjut hingga software benar-benar terbebas dari bug, bekerja stabil, dan berfungsi sesuai harapan.

f. Implementation dan Release

Setelah fase pengujian perangkat lunak selesai dan tidak ada bug yang tersisa pada sistem, maka tahap implementasi dapat dimulai. Tahap ini biasanya juga disebut sebagai tahap deployment. Tujuan dari tahap ini adalah untuk men-deploy perangkat lunak ke lingkungan produksi sehingga users dapat mulai menggunakannya. Fase ini melibatkan penginstalan aktual dari sistem yang baru dikembangkan. Untuk proyek sederhana, contoh deployment seperti menerapkan kode ke server web. Sedangkan untuk proyek pengembagan software berskala besar, deployment akan melibatkan proses integrasi dengan banyak sistem berbeda. Meskipun demikian, banyak perusahaan memilih agar produk akhir dapat pertama kali dirilis dalam segmen terbatas dan diuji di lingkungan bisnis (UAT-User Acceptance Testing) sebelum benar-benar dirilis ke pasar. Hal ini juga dilakukan untuk meminimalisir adanya masalah yang ditemukan oleh users setelah produk dirilis ke pasar.

g. Maintenance

Tahapan SDLC yang terakhir adalah proses maintenance atau pemeliharaan software. Di tahap ini, tim akan melakukan pemeliharaan sistem dan rutin melakukan pembaruan agar kinerja software tetap dapat optimal. Biasanya beberapa aktivitas maintenance yang dilakukan adalah:

- Perbaikan bug: perbaikan bug ketika ada masalah yang dilaporkan.
- Upgrade sistem : meningkatkan kinerja software dengan sistem yang lebih baru.
- Peningkatan fitur: menambahkan fitur atau fungsionalitas pada pada software yang dikembangkan.

Model-Model SDLC

Dalam proses pengembangan software, terdapat beberapa model SDLC yang populer digunakan. Berikut beberapa diantaranya:

a. Waterfall

Waterfall adalah salah satu model SDLC tertua. Kerangka kerja ini menekankan untuk terus maju dari satu tahap ke tahap berikutnya. Jadi, Anda harus benar-benar menyelesaikan suatu tahap secara sepenuhnya sebelum melanjutkan ke tahap berikutnya. Kerangka kerja ini cocok untuk proyek kecil dengan hasil akhir yang mudah ditentukan dari awal. Namun untuk pengerjaan proyek besar, sebaiknya hindari penggunaan Waterfall karena kerangka kerja ini tidak cocok untuk proyek yang kompleks serta tidak fleksibel terhadap perubahan.

b. Agile

Agile adalah kerangka kerja untuk pengembangan software dengan proses yang ramping namun dapat menghasilkan produk akhir berkualitas tinggi. Kerangka kerja ini akan menggunakan urutan kerja inkremental (berkembang sedikit demi sedikit secara teratur) dan iteratif (berulang). Model SDLC ini membutuhkan tim pengembang yang dapat beradaptasi dengan cepat terhadap perubahan dan cocok untuk pengembangan proyek berskala besar. Beberapa kelebihan dari kerangka kerja ini seperti memungkinkan pengembangan dan pengujian yang cepat, masalah atau bug dapat segera terdeteksi dan diperbaiki, serta mengakomodasi perubahan atau peningkatan produk dalam proses pengembangan.

c. DevOps

DevOps merupakan gabungan dari dua kata yaitu Development dan Operations. Kerangka kerja ini adalah kombinasi dari culture, praktik, serta tools untuk meningkatkan kemampuan perusahaan dalam proses delivery produk dengan kecepatan tinggi. Secara garis besar, DevOps akan membentuk kolaborasi yang lebih erat antara tim pengembangan dan tim operasi untuk pengembangan produk. Cara ini dapat membantu perusahaan untuk menyelaraskan orang-orang, proses, dan alat yang digunakan untuk mencapai kepuasan pelanggan.

Perancangan Perangkat Lunak

Definisi : Perancangan perangkt lunak adalah disiplin manajerial dan teknis yang berkaitan dengan pembuatan dan pemeliharaan produk perangkat lunak secara sistematis, termasuk pengembangan dan modifikasinya, yang dilakukan pada waktu yang tepat dan dengan mempertimbangkan faktor biaya.

Tujuan : Memperbaiki kualitas produk perangkat lunak, meningkatkan produktivitas.

Pengertian produk perangkat lunak : Perangkat lunak yang digunakan oleh berbagai pengguna.

Hal-hal yang perlu diperhatikan dalam pengembangan sebuah produk perangkat lunak : Kebutuhan dan batasan-batasan pengguna, mengakomodasi paling tidak kepentingan tiga pihak, tahap uji coba, dokumen pendukung, pelatihan.

Beberapa atribut yang merupakan ukuran kualitas perangkat lunak :

- Maintanability (rawatan), harus dapat dengan mudah dirubah dengan perubahan kebutuhan pengguna.
- Dependability (ketergantungan) , harus dapat dipercaya (trustworthy) sehingga pengguna dapat menggantungkan sepenuhnya proses bisnis mereka.
- Eciency, harus esien dan tidak memakai resources yang tinggi.
- Usability, perangkat Lunak harus dapat digunakan (useble) oleh penggunanya dalam memenuhi kebutuhan mereka.

Distribusi Upaya

- Masa hidup sebuah produk perangkat lunak adalah 1 sd 3 tahun dalam pengembangan dan 5 sd 15 tahun dalam pemakaiannya (pemeliharaannya).
- Distribusi upaya antara pengembangan dan pemeliharaan bervariasi antara 40/60, 30/70 dan bahkan 10/90.
- Tiga aktivitas pengembangan perangkat lunak adalah : analisa dan perancangan, implementasi dan pengujian.
- Tiga aktivitas pemeliharaan perangkat lunak adalah : peningkatan kemampupan produk, penyesuaian produk dengan lingkungan pemroses baru dan perbaikan.

Apa yg disebut perangkat lunak?

- Instruksi (Program Komputer) : yang bila dieksekusi dapat

menjalankan fungsi tertentu. Struktur data : yang dapat membuat program memanipulasi informasi. Dokumen : Yang menjelaskan operasi dan penggunaan program (Roger Pressman).

- Program Komputer, Prosedur, aturan dan dokumentasi : yang berkaitan dengannya serta data yang berkaitan dengan operasi suatu sistem komputer (IEEE).

Jenis-jenis Perangakt Lunak :

- Perangkat Lunak Sistem (system software)
- Perangkat Lunak Waktu Nyata (real-time software)
- Perangakat Lunak Bisnis (Business software).
- Perangkat Lunak Rekayasa dan Ilmu Pengetahuan (engineering and scientific software).
- Embedded software
- Perangkat Lunak Pribadi (Personal Software)
- Perangkat Lunak Intelegensia buatan (artificial inteligent software).
- Perangakat Lunak lainnya.

Dokumen Perangkat Lunak :

- Software Project Management Plan (SPMP) / Rencana Manajemen Proyek Perangkat Lunak
- Software Requirement Specification (SRS) / Spesifikasi Kebutuhan Software
- Software Design Description (SDD) / Deskripsi Desain Software Software Test Plan (STP) / Rencana Uji Software.
- Software Test Description (STD) / Deskripsi Uji Software.
- Software Test Result (STR) / Hasil Uji Software.
- Software Version / Versi Software.
- User Guide / User Manual.

Pengembangan Perangkat Lunak

Pengembangan Perangkat Lunak :
Proses dimana persoalan/kebutuhan pemakai diterjemahkan menjadi produk perangkat lunak melalui suatu rangkaian aktivitas tertentu sesuai model proses, metode dan alat bantu yang digunakan.
Elemen-elemen proses pengembangan : Model Proses Pengembangan :

- Cara atau strategi bagaimana perangkat lunak dibuat sedemikian rupa sehingga produk perangkat lunak tersebut dapat diwujudkan.
- Beberapa model proses pengembangan perangkat lunak :
    1. Waterfall
    2. Incremental
    3. Prototyping Model
    4. Spiral Model
    5. Rational Unified Process (RUP)
    6. Extreme Programming (XP)

Kesimpulan

Pengujian perangkat lunak (bahasa Inggris: software testing) merupakan suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (under test).[1] Pengujian perangkat lunak juga memberikan pandangan mengenai perangkat lunak secara obyektif dan independen, yang bermanfaat dalam operasional bisnis untuk memahami tingkat risiko pada implementasinya. Teknik-teknik pengujian mencakup, tetapi tidak terbatas pada, proses mengeksekusi suatu bagian program atau keseluruhan aplikasi dengan tujuan untuk menemukan bug perangkat lunak (kesalahan atau cacat lainnya).Lainnya [2] Pengujian perangkat lunak dapat dinyatakan sebagai proses validasi dan verifikasi bahwa sebuah

program / aplikasi / produk:

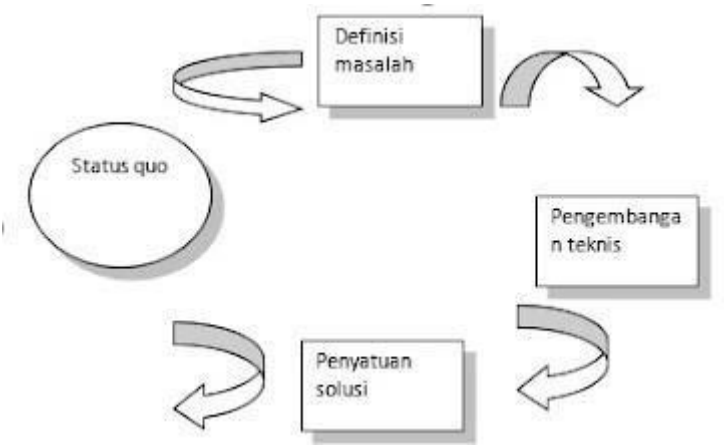
1. Memenuhi kebutuhan (requirement) yang mendasari perancangan dan pengembangan perangkat lunak tersebut;
2. Berjalan sesuai dengan yang diharapkan;
3. Dapat diterapkan menggunakan karakteristik yang sama;
4. Memenuhi kebutuhan semua pihak yang berkepentingan.

PENGEMBANGAN SOFTWARE, TESTING DAN EVOLUSI

Pengertian Topologi Jaringan

Model Sekuensial Linier (Waterfall)

Model Sekuensial Linier (Waterfall)



Gambar 1. 1 Model Sekuensial Linier

Model Sekuensial Linier atau Waterfall adalah sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan.

Tahapan – tahapan dalam pengembangan model sekuensial linear / Waterfall sebagai berikut:

1. Rekayasa dan Pemodelan Sistem / Informasi

Karena perangkat lunak selalu merupakan bagian dari sebuah sistem (bisnis) yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke perangkat lunak tersebut. Pandangan sistem ini penting ketika perangkat lunak harus berhubungan dengan elemen-elemen yang lain seperti perangkat lunak, manusia dan database.

2. Analisis kebutuhan perangkat lunak

Proses pengumpulan kebutuhan difokuskan khususnya untuk perangkat lunak,perekayasa perangkat lunak (Analis) harus memahami domain permasalahan (problem domain), tingkah laku, unjuk kerja dan antarmuka (interface) yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.

3. Generasi kode

Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode meliputi pekerjaan dalam langkah ini, dan dapat dilakukan secara mekanis.

4. Desain

Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda (struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural. Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan

kode (coding). Sebagaimana analisis, desain ini juga didokumentasikan.

5. Pengujian (Tes)

Sekali kode dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan seudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

6. Pemeliharaan

Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah perangkat lunak yang dilekatkan). Perubahan akan terjadi karena kesalahan kesalahan karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahan-perubahan di dalam lingkungan eksternalnya atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan perangkat lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

Model Waterfall adalah model yang paling lama dan banyak secara luas digunakan sebagai paradigma untuk rekayasa software.

1. Kelebihan :
2. Sederhana.
3. Langkah-secara terurut.
4. Fokus.
5. Mudah diikuti.

Kekurangan:

1. Tidak flesible sebab proyek yang nyata jarang mengikuti aliran yang terurut untuk menyusun Model.

2. Sangat sulit untuk customer terhadap mengahadapi semua kebutuhan yang eksplisit.
3. Customer harus memiliki kesabaran untuk menunggu keabsahan produk software dalam fase
4. Yang terlambat.(sampai program dimplementasikan)
5. Customer tercakup dalam awal proyek.
6. Pembuat sering ditunda secara tidak berhubungan diantara fase.

Model Prototipe

Gambar 1. 2 Model Prototipe

Metode Prototype merupakan suatu paradigma baru dalam metode pengembangan perangkat lunak dimana metode ini tidak hanya sekedar evolusi dalam dunia pengembangan perangkat lunak, tetapi juga merevolusi metode pengembangan perangkat lunak yang lama yaitu sistem sekuensial yang biasa dikenal dengan nama SDLC atau waterfall development model.

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Prototipe bisa

menjadi paradigma yang efektif bagi rekayasa perangkat lunak. Kuncinya adalah mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang keduanya harus setuju bahwa prototype dibangun untuk berfungsi sebagai mekanisme pendefinisian kebutuhan. Prototype kemudian disingkirkan dan perangkat lunak actual direkayasa dengan tertuju kepada kualitas dan kemampuan pemeliharaan.

Berikut adalah Tahapan – tahapan Proses Pengembangan dalam Model Prototype, yaitu :

1. Pengumpulan kebutuhan Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.
2. Membangun prototyping Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output).
3. Evaluasi protoptyping Evaluasi ini dilakukan oleh pelanggan, apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan pelanggan atau belum. Jika sudah sesuai, maka langkah selanjutnya akan diambil. Namun jika tidak, prototyping direvisi dengan mengulang langkah-langkah sebelumnya.
4. Mengkodekan sistem Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Menguji sistem Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, kemudian dilakukan proses Pengujian. Pengujian ini dilakukan dengan White Box, Black Box, Basis Path, pengujian arsitektur, dll.
6. Evaluasi Sistem Pelanggan mengevaluasi apakah perangkat lunak yang sudah jadi sudah sesuai dengan yang diharapkan . Jika ya, maka proses akan dilanjutkan ke tahap selanjutnya, namun jika perangkat lunak yang sudah jadi tidak/belum sesuai dengan apa yang diharapkan, maka tahapan sebelumnya akan diulang.
7. Menggunakan sistem Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

Model Prototyping ini sangat sesuai diterapkan untuk kondisi yang beresiko tinggi di mana masalah-masalah tidak terstruktur dengan baik, terdapat fluktuasi kebutuhan pemakai yang berubah dari waktu ke waktu atau yang tidak terduga, bila interaksi dengan pemakai menjadi syarat mutlak dan waktu yang tersedia sangat terbatas sehingga butuh penyelesaian yang segera. Model ini juga dapat berjalan dengan maksimal pada situasi di mana sistem yang diharapkan adalah yang inovatif dan mutakhir sementara tahap penggunaan sistemnya relatif singkat.

Berikut merupakan Jenis – jenis dari Prototyping :

1. Feasibility prototyping Digunakan untuk menguji kelayakan dari teknologi yang akan digunakan untuk system informasi yang akan disusun.
2. Requirement prototyping Digunakan untuk mengetahui kebutuhan aktivitas bisnis user.
3. Desain Prototyping Digunakan untuk mendorong perancangan sistem informasi yang akan digunakan.

4. Implementation prototyping Merupakan lanjutan dari rancangan prototype, prototype ini langsung disusun sebagai suatu sistem informasi yang akan digunakan.

Kelebihan:

1. Mudah dan cepat identifikasi kebutuhan customer
2. Customer mengecek protipe di awal tingkatan dan menyediakan input dan umpan baliknya.
3. Persetujuan yang baik dengan mengikuti kasus:
   a. Customer tidak bisa menyediakn kebutuhan yang jelas.
   b. Sangat rumit interaksi sistem dari pengguna
   c. Menggunakan teknologi baru, hardware dan algoritma
   d. Membuat domain baru sistem aplikasi.

Kelemahan :

1. Prototipe bisa melayani sebagai “sistem pertama”.Brooks merekomendasikan untuk membuangnya.
2. Developer biasa ikut membuat produk didasarkan pada prototipe.
3. Developer sering membuat perjanjian implementasi dalam memerintahkan untuk dapat prototipe bekerja secara cepat.
4. Customer boleh terkejut bahwaa protipe adalah tidak sebuah produk, yang dibuat dengannya.

Model RAD

Rapid Application Development (RAD) adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi “kecepatan tinggi” dari model sekuensial linier dimana perkembangan cepat dicapai dengan

menggunakan pendekatan konstruksi berbasis komponen. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan “sistem fungsional yang utuh” dalam periode waktu yang sangat pendek (kira-kira 60-90 hari).

Pendekatan RAD melingkupi fase-fase sebagai berikut :

1. Pemodelan Bisnis Aliran informasi diantara fungsi-fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan-pertanyaan berikut : Informasi apa yang mengendalikan proses bisnis? Informasi apa yang dimunculkan? Siapa yang memunculkannya? Kemana informasi itu pergi? Siapa yang memprosesnya?
2. Pemodelan Data. Aliran informasi yang didefinisikan sebagai bagian dari fase business modelling disaring kedalam serangkaian objek data yang dibutuhkan untuk mendukung bisnis tersebut.
3. Pemodelan Proses Aliran informasi yang didefinisikan di dalam fase data modeling ditransfirmasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.
4. Pembentukan Aplikasi RAD mengasumsikan pemakaian teknik generasi keempat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang ada atau menciptakan komponen yang bisa dipakai lagi.
5. Pengujian dan Turnover. Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tetapi komponen baru harus diuji dan semua interface harus dilatih secara penuh.

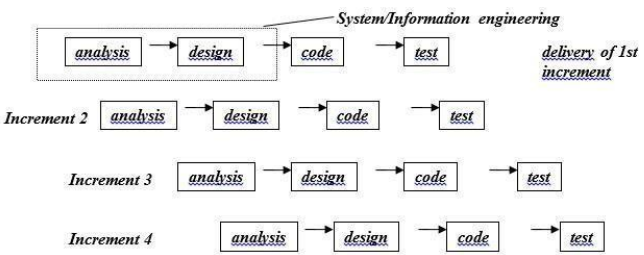
Kelebihan :

1. Waktu pembuatan yang pendek.
2. Pengurangan biaya supaya software digunakan kemabali dan konstruksi dasar komponen.

Kelemahan:

1. Untuk yang besar, tetapi proyek yang berskala, RAD butuh sumber yang cukup.
2. RAD butuh developer dan pelanggan yang diijinkan untuk menyusun.
3. Pembuatan software adalah spesifik proyek, dan tidak boleh dimodulkan secara baik.
4. Kualitasnya tergantung pada kualitas dari komponen yang ada.
5. Proyek yang tidak akurat dengan resiko teknik yang tinggi dan teknologi

Model Incremental



Gambar 1. 3 Model Incremental

Merupakan kombinasi linear sequential model (diaplikasikan secara berulang) dan filosofi pengulangan dari prototyping model. Setiap tahapan linear sequential menghasilkan deliverable increment bagi perangkat lunak, dimana increment pertamanya merupakan sebuah produk inti yang mewakili kebutuhan dasar sistem. Produk inti ini nantinya dikembangkan menjadi incrementincrement selanjutnya setelah digunakan dan dievaluasi sampai didapat produk yang lengkap dan memenuhi kebutuhan pemakai.

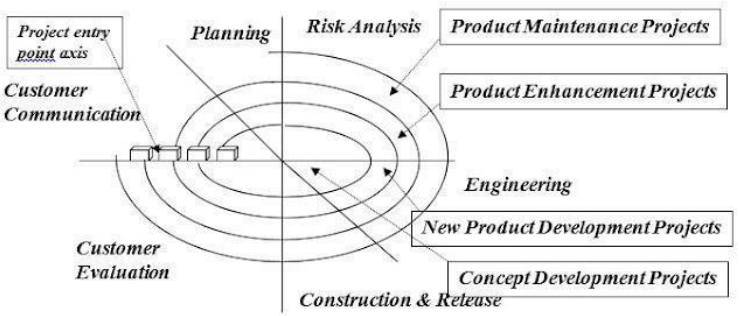
Kelebihan :

1. Personil bekerja optimal.
2. Mampu mengakomodasi perubahan secara fleksibel, dengan waktu yang relatif singkat dan tidak dibutuhkan anggota/tim kerja yang banyak untuk menjalankannya.
3. Pihak konsumen dapat langsung menggunakan dahulu bagian-bagian yang telah selesai dibangun. Contohnya pemasukan data karyawan.
4. Mengurangi trauma karena perubahan sistem. Klien dibiasakan perlahan-lahan menggunakan produknya setiap bagian demi bagian. Memaksimalkan pengembalian modal investasi konsumen.

Kekurangan :

1. Tidak cocok untuk proyek berukuran besar (lebih dari 200.000 baris coding).
2. Sulit untuk memetakan kebutuhan pemakai ke dalam rencana spesifikasi tiap-tiap hasil dari increament.

Model Spiral



Gambar 1. 4 Model Spiral

Merupakan model proses perangkat lunak yang memadukan wujud

pengulangan dari model prototyping dengan aspek pengendalian dan sistematika dari linear sequential model. Dalam model ini perangkat lunak dikembangkan dalam suatu seri incremental release.

Spiral model dibagi menjadi 6 aktivitas kerangka kerja sebagai berikut:

1. Komunikasi dengan pemakai
2. Perencanaan
3. Analsis resiko
4. Rekayasa
5. Konstruksi dan pelepasan
6. Evaluasi

Tahap-tahap model ini dapat dijelaskan secara ringkas sebagai berikut :

1. Tahap Liason
   Pada tahap ini dibangun komunikasi yang baik dengan calon pengguna/pemakai.
2. Tahap Planning (perencanaan)
   Pada tahap ini ditentukan sumber-sumber informasi, batas waktu dan informasi-informasi yang dapat menjelaskan proyek.
3. Tahap Analisis Resiko:
   Mendefinisikan resiko, menentukan apa saja yang menjadi resiko baik teknis maupun manajemen.
4. Tahap Rekayasa (engineering)
   Pembuatan prototipe
5. Tahap Konstruksi dan Pelepasan (release)
   Pada tahap ini dilakukan pembangunan perangkat lunak yang dimaksud, diuji, diinstal dan diberikan sokongan-sokongan tambahan untuk keberhasilan proyek.
6. Tahap Evaluasi Pelanggan/pemakai/pengguna

Biasanya memberikan masukan berdasarkan hasil yang didapat dari tahap engineering dan instalasi.

Kelebihan :
Model ini sangat mempertimbangkan resiko kemungkinan munculnya kesalahan sehingga sangat dapat diandalkan untuk pengembangan perangkat lunak skala besar. Pendekatan model ini dilakukan melalui tahapan-tahapan yang sangat baik dengan menggabungkan model waterfall ditambah dengan pengulangan-pengulangan sehingga lebih realistis untuk mencerminkan keadaan sebenarnya. Baik pengembang maupun pemakai dapat cepat mengetahui letak kekurangan dan kesalahan dari sistem karena proses-prosesnya dapat diamati dengan baik.

Kekurangan :
Waktu yang dibutuhkan untuk mengembangkan perangkat lunak cukup panjang demikian juga biaya yang besar. Selain itu, sangat tergantung kepada tenaga ahli yang dapat memperkirakan resiko. Terdapat pula kesulitan untuk mengontrol proses. Sampai saat ini, karena masih relatif baru, belum ada bukti apakah metode ini cukup handal untuk diterapkan.

Kesimpulan

Proses Pengembangan Perangkat Lunak (Software Development Process) adalah suatu penerapan struktur pada pengembangan suatu Perangkat Lunak (Software), yang bertujuan untuk mengembangkan sistem dan memberikan panduan untuk menyukseskan proyek pengembangan sistem melalui tahapan-tahapan tertentu. Dalam prosesnya, terdapat beberapa paradigma model pengembangan sistem perangkat lunak.

Pendekatan Strategis Pengujian PL

Software Testing (Pengujian PerangkatLunak)

adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Pentingnya pengujian perangkat lunak dan implikasinya yang mengacu pada kualitas perangkat lunak tidak dapat terlalu ditekan karena melibatkan sederetan aktivitas produksi di mana peluang terjadinya kesalahan manusia sangat besar dan arena ketidakmampuan manusia untuk melakukan dan berkomunikasi dengan sempurna maka pengembangan perangkat lunak diiringi dengan aktivitas jaminan kualitas.

Gambar 2. 1 Software Testing

Meningkatnya visibilitas (kemampuan) perangkat lunak sebagai suatu elemen sistem dan “biaya” yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti. Pada dasarnya, pengujian merupakan satu langkah dalam proses rekayasa perangkat lunak yang dapat dianggap sebagai hal yang merusak daripada membangun.

Meningkatnya visibilitas (kemampuan) perangkat lunak sebagai suatu elemen sistem dan “biaya” yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti. Pada dasarnya, pengujian merupakan satu langkah dalam proses rekayasa perangkat lunak yang dapat dianggap sebagai hal yang merusak daripada membangun.

Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah:

Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan

Test case yang baik
adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.

Pengujian yang sukses
adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya Sasaran itu berlawanan dengan pandangan yang biasanya dipegang yang menyatakan bahwa pengujian yang berhasil adalah pengujian yang tidak ada kesalahan yang ditemukan.

Data yang dikumpulkan pada saat pengujian dilakukan memberikan indikasi yang baik mengenai reliabilitas perangkat lunak dan beberapa menunjukkan kualitas perangkat lunak secara keseluruhan, tetapi ada satu hal yang tidak dapat dilakukan oleh pengujian, yaitu pengujian tidak dapat memperlihatkan tidak adanya cacat, pengujian hanya dapat memperlihatkan bahwa ada kesalahan perangkat lunak. Sebelum mengaplikasikan metode untuk mendesain test case yang efektif, perekayasa perangkat lunak harus memahami prinsip dasar yang menuntun pengujian perangkat lunak, yaitu:

Pengujian white-box

Berfokus pada struktur control program. Test case dilakukan untuk memastikan bahwa semua statemen pada program telah dieksekusi paling tidak satu kali selama pengujian dan bahwa semua kondisi logis telah diuji. Pengujian basic path, tehnik pengujian white-box, menggunakan grafik (matriks grafiks) untuk melakukan serangkaian pengujian yang independent secara linear yang akan memastikan cakupan. Pengujian aliran data dan kondisi lebih lanjut menggunakan logika program dan pengujian loop menyempurnakan tehnik white-box yang lain dengan memberikan sebuah prosedur untuk menguji loop dari tingkat kompleksitas yang bervariasi. Pengujian black-box didesain untuk mengungkap kesalahan pada persyaratan fungsional tanpa mengabaikan kerja internal dari suatu program.

-->semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan, maksudnya mengungkap kesalahan dari cacat yang menyebabkan program gagal.
-->Pengujian harus direncanakan lama sebelum pengujian itu mulai, maksudnya semua pengujian dapat direncanakan dan dirancang sebelum semua kode dijalankan
-->Prinsip Pareto berlaku untuk pengujian perangkat lunak, maksudnya dari 80% kesalahan yang ditemukan selamapengujian dapat ditelusuri sampai 20% dari semua modul program.
-->Pengujian harus mulai “dari yang kecil” dan berkembang ke pengujian “yang besar”, Selagi pengujian berlangsung maju, pengujian mengubah focus dalam usaha menemukan kesalahan pada cluster modul yang terintegrasi dan akhirnya pada sistem.
-->Pengujian yang mendalam tidak mungkin karena tidak mungkin mengeksekusi setiap kombinasi jalur skema pengujian dikarenakan jumlah jalur permutasi untuk program menengahpun sangat besar.
-->Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang independent.

Dalam lingkungan yang ideal, perekayasa perangkat lunak mendesain suatu program computer, sebuah sistem atau produk dengan testabilitas dalam pikirannya. Hal ini memungkinkan individu yang berurusan dengan pengujian mendesain test case yang efektif secara lebih mudah. Testabilitas adalah seberapa mudah sebuah program computer dapat diuji. Karena sangat sulit, perlu diketahui apa yang dapat dilakukan untuk membuatnya menjadi lebih mudah. Procedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. Sasaran utama desain test case adalah untuk mendapatkan serangkaian pengujian yang memiliki kemungkinan tertinggi di dalam pengungkapan kesalahan pada perangkat lunak. Untuk mencapai sasaran tersebut, digunakan 4 kategori yang berbeda dari tehnik desain test case: Pengujian whitebox, pengujian black-box, Integrasi Bottom-Up dan Integrasi Top-Down.

Tehnik pengujian black-box

Berfokus pada domain informasi dari perangkat lunak, dengan melakukan test case dengan menpartisi domain input dari suatu program dengan cara yang memberikan cakupan pengujian yang mendalam. Metode pengujian graph-based mengeksplorasi hubungan antara dan tingkah laku objek-objek program. Partisi ekivalensi membagi domain input ke dalam kelas data yang mungkin untuk melakukan fungsi perangkat lunak tertentu. Analisis nilai batas memeriksaa kemampuan program untuk menangani data pada batas yang dapat diterima. Metode pengujian yang terspesialisasi meliputi sejumlah luas kemampuan perangkat lunak dan area aplikasi. GUI, arsitektur client/ server, dokumentasi dan fasilitas help dan sistem real time masing-masing membutuhkan

pedoman dan tehnik khusus untuk pengujian perangkat lunak.

Integrasi Top-Down

Adalah pendekatan incremental dengan menggerakkan ke bawah melalui hirarki control, dimulai dengan control utama. Strategi intergrasi top-down memeriksa control mayor atau keputusan pada saat awal di dalam proses pengujian. Pada struktur program yang difaktorkan dengan baik, penarikan keputusan terjadi pada tingkat hirarki yang lebih tinggi sehingga terjadi lebih dulu. Strategi top-down kelihatannya tidak sangat rumit, tetapi di dalam praktenya banyak menimbulkan masalah logistic. Biasanya masalah ini terjadi jika dibutuhkan pemrosesan di dalam hirarki pada tingkat rendah untuk menguji secara memadai tingkat yang lebih tinggi.

Pengujian Integrasi Bottom-up

Memulai konstruksi dan pengujian dengan modul atomic (modul pada tingkat paling rendah pada struktur program). Karena modul diintegrasikan dari bawah ke atas, maka pemrosesan yang diperlukan untuk modul subordinate ke suatu tuingkat yang diberikan akan selalu tersedia dan kebutuhan akan stub dapat dieliminasi. Strategi integrasi bottom-up dapat diimplementasi dengan langkah-langkah: Modul tingkat rendah digabung ke dalam cluster (build) yang melakukan subfungsi perangkat lunak spesifik. Driver (program control untuk pengujian) ditulis untuk mengkoordinasi input dan output test case Cluster diuji Driver diganti dan cluster digabungkan dengan menggerakkannya ke atas di dalam struktur program.

Kesimpulan

Pengujian Perangkat Lunak adalah proses menjalankan dan mengevaluasi sebuah PL secara manual maupun otomatis untuk menguji apakah PL sudah memenuhi persyaratan atau belum atau untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya. Pelaksanaan pengujian PL biasanya disesuaikan dengan metodologi pembangunan PL yang digunakan. Pada umumnya pengujian dilakukan sesudah tahap pemograman,namun demikian perencanaan pengujian dilakukan mulai tahap analisis.

Tujuan Pengujian

1. Menilai apakah perangkat lunak yang dikembangkan telah memenuhi kebutuhan pemakai.
2. Menilai apakah tahap pengembangn PL sesuai dengan metodologi yang digunakan.
3. Membuat dokumentasi hasil pengujian yang menginformasikan kesesuaian PL yang diuji dengan spesifikasi yang ditentukan.

Tahap Pengujian

1. Tentukan apa yang akan diukur melalui pengujian.
2. Bagaimana pengujian akan dilaksanakan.
3. Membangun suatu kasus uji (test case) yaitu sekumpulan data atau situasi yang akan digunakan dalam pengujian.
4. Tentukan hasil yang akan diharapkan atau hasil yang sebenarnya.
5. Jalankan Kasus pengujian.
6. Bandingkan hasil pengujian dan hasil yang diharapkan.

Pengujian Tahap Analisis Pengujian pada tahap analisis ditekankan pada validasi terhadap kebutuhan,untuk menjamin bahwa kebutuhan telah dispesifikasi dengan benar. Tujuan pengujian pada tahap ini adalah untuk mendapatkan kebutuhan yang layak dan untuk memastikan apakah kebutuhan tersebut sudah dirumuskan dengan baik.

Faktor-faktor pengujian yang dilakukan pada tahap analisis :

1. Kebutuhan yang berkaitan dengan metodologi.
2. Pendefenisian spesifikasi fungsional
3. Penentuan spesifikasi kegunaan.
4. Penentuan kebutuhan portabilitas.
5. Pendefenisian antar muka sistem. Pengujian Tahap Perancangan

Pengujian tahap perancangan bertujuan untuk menguji struktur perangkat lunak yang diturunkan dari kebutuhan. Kebutuhan yang bersifat umum dirinci menjadi bentuk yang lebih spesifik.

Faktor-faktor pengujian yang dilakukan pada tahap perancangan :
Perancangan yang berkaitan dengan kebutuhan.

1. Kesesuaian perancangan dengan metodologi dan teori.
2. Portabilitas rancangan.
3. Perancangan perawatan.
4. Kebenaran rancangan berkaitan denganfungsi dan aliran data.
5. Kelengkapan perancangan antar muka.

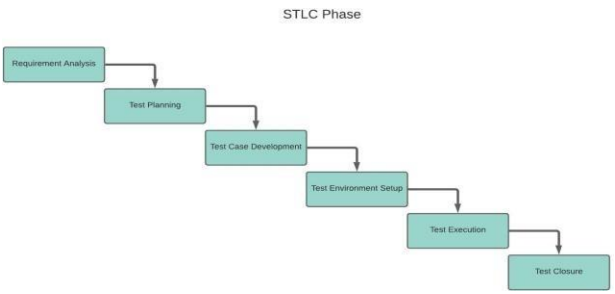
RENCANAPENGUJIAN IMPLEMENTASI DAN DOKUMENTASI HASIL PENGUJIAN

Software Testing Life Cycle (STLC)

Bagi orang-orang yang bergelut dalam sebuah pengembangan software maka pasti sudah mengetahui tentang yang namanya SDLC, kepanjangan dari Software Development Life Cycle. Namun apakah sudah mengetahui atau pernah mendengar tentang STLC? STLC mungkin sudah tidak asing di telinga Quality Assurance karena memang STLC berfokus pada tahapan testing. Software Testing Life Cycle (STLC) adalah serangkaian aktivitas sistematis dan terencana yang dilakukan selama melakukan software testing guna memastikan capaian kualitas software terpenuhi. STLC memiliki fase yang setiap fasenya memiliki tujuan dan hasil.

Fase STLC mungkin akan memiliki banyak versi, karena menyesuaikan kebutuhan dari perusahaan. Namun, STLC versi apapun pasti memiliki dasar fase STCL yang sama.

Fase yang ada di STLC, yaitu:



Gambar 3. 1 Fase STLC

Requirement Analysis

Mempelajari dan menganalisa kebutuhan dari perspektif pengujian untuk mengidentifikasi apakah kebutuhan dapat dilakukan testing atau tidak. Jika kebutuhan tidak dapat dilakukan testing, maka lakukan komunikasi ke tim agar dapat direncanakan strategi lain untuk mengurangi rabbit hole atau hal yang membuat stuck di akhir. Tim QA yang masih merasa belum paham akan kebutuhan testing juga dapat berkomunikasi aktif dengan stakeholder terkait (Business Analyst, Project Leader, Client dll) untuk dapat memahami kebutuhan secara detail. Cakupan testing juga dideklarasikan di fase ini.

Entry Criteria

- Dokumen requirements
- Arsitektur aplikasi
- Acceptence criteria

Activities

- Menyiapkan pertanyaan untuk dijawab oleh stakeholder
- Mengidentifikasi tipe testing yang perlu dilakukan (Functional, Security, Performance dll)

- Menjabarkan fokus dan prioritas testing
- Menyiapkan RTM (Requirements Traceability Matrix)
- Mengidentifikasi detail environment di mana testing akan dilakukan.
- Melakukan pengecekan kelayakan automasi jika dibutuhkan dan menyiapkan laporan kelayakan automasi Deliverables.
- Daftar pertanyaan dan jawaban yang sudah terjawab.
- Dokumen RTM (Requirements Traceability Matrix).
- Laporan kelayakan automasi.

Deliverables

- Dokumen test plan atau test strategy
- Dokumen perkiraan usaha testing

Test Case Development

Tim QA akan melakukan pembuatan test case secara rinci dan juga menyiapkan test data yang akan digunakaan pada testing nantinya. Setelah test case sudah selesai, maka akan dilakukan review oleh rekan lainnya atau lead test.

Entry Criteria

- Dokumen requirements
- Dokumen RTM (Requirements Traceability Matrix) dan test plan
- Laporan analisis automasi

Activities

- Menyiapkan dokumentasi test case
- Menyiapkan script automation test
- Melakukan review test case dan script test • Menyiapkan kembali data yang akan digunakan untuk testing

Deliverables

- Dokumen test cases
- Test script
- Test data yang akan digunakan

Test Environment Setup

Fase ini adalah fase yang tidak bergantung dengan yang lainnya, sehingga dapat dikerjakan bersamaan dengan fase Test Case Development. Fase ini bisa jadi tidak dilakukan oleh tim QA, jika tim developer nantinya yang akan menyiapkan.

Entry Criteria

- Test plan dan test data
- Smoke test cases
- Dokumen desain dan arsitektur sistem
- Dokumen penyiapan environment

Activities

- Menganalisa perangkat lunak dan perangkat keras yang dibutuhkan
- Menyiapkan test environment
- Melakukan smoke testing

Deliverables

- Hasil dari smoke testing
- Environment siap untuk diinputkan dengan test data

Test Execution

Fase ini adalah fase puncaknya yaitu melakukan testing. Pada fase ini akan dilakukan testing berdasarkan test plan dan test case yang sudah disiapkan di fase sebelumnya. Tim QA akan melakukan testing dan menandai tiap case-nya apakah lulus atau gagal.

Entry Criteria

- Dokumen test plan atau test strategy
- Test script
- Test case dan test data
- Test environment siap digunakan

Activities

- Mengeksekusi test case sesuai test planning yang sudah dibuat sebelumnya
- Melakukan dokumentasi dan memberikan tanda pada cases yang berhasil dan gagal
- Menugaskan perbaikan defect kepada programmer
- Melakukan testing ulang untuk defect yang sudah diperbaiki
- Melacak defect untuk dapat ditutup

Deliverables

- Defect report
- Test cases yang sudah diperbarui dengan hasil test-nya
- Melengkapi dokumen RTM dengan status eksekusi

Test Closure

Semua anggota tim QA akan berkumpul bersama untuk melakukan diskusi dan melaporkan hasil pengujian yang sudah dilakukan. Pada diskusi dapat membahas terkait hambatan ketika testing, apa yang dapat ditingkatkan dan lain-lain. Untuk defect yang ditemukan juga dilakukan analisa sesuai dengan priority dan severity-nya.

Entry Criteria

- Semua test case sudah selesai dieksekusi
- Laporan eksekusi test case
- Laporan defect

Activities

- Mengevaluasi penyelesaian siklus berdasarkan waktu, test scope, biaya, perangkat lunak,
- kualitas, dan kepentingan tujuan bisnis.
- Menyiapkan laporan test closure dan metrik pengujian
- Dokumentasi lesson learn dari proyek tersebut
- Laporan kualitatif dan kuantitatif kualitas produk kepada konsumen
- Analisa hasil testing untuk mendistirbusikan defect berdasarkan priority dan severity (baca: Tipe Bug pada Software Testing)

Deliverables

- Laporan test closure
- Metriks pengujian

Kesimpulan

Software Testing Life Cycle (STLC) adalah tahap-tahap proses pengujian yang dilaksanakan secara sistematis dan terencana. Dalam proses STLC, berbagai kegiatan dilakukan untuk meningkatkan kualitas produk. Meskipun SDLC dan STLC terkesan mirip, namun penerapannya berbeda. SDLC diterapkan pada semua tahap pengembangan software, sedangkan STLC hanya terbatas pada tahap pengujian software.

Rencana Pengujian, Implementasi, dan Dokumen Hasil Pengujian Lanjutan

Object-oriented testing:

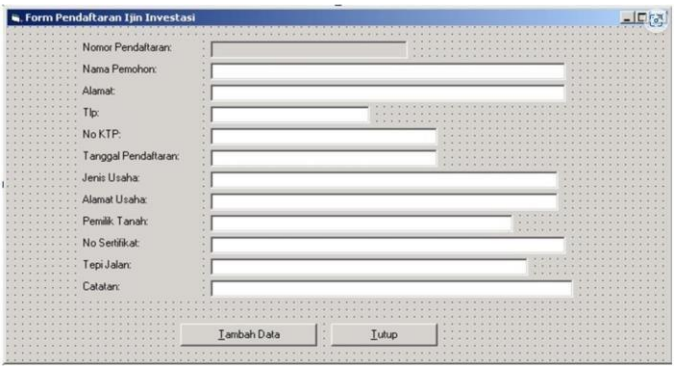
Object-oriented testing Object-oriented testing

1. Komponen yang diuji adalah class-object.
2. Lebih besar dibandingkan pengujian suatu function sehingga pendekatan white-box testing perlu diperluas.
3. Tidak jelasnya 'top' suatu system untuk top-down integration dan testing.

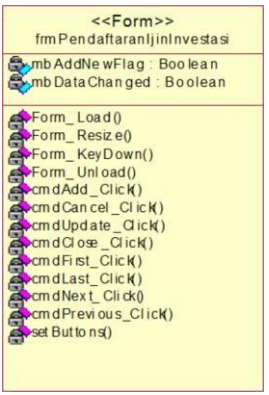
Testing levels

1. Testing operations pada objects
2. Testing object classes
3. Testing clusters cooperating objects
4. Testing OO system secara lengkap

Object Form



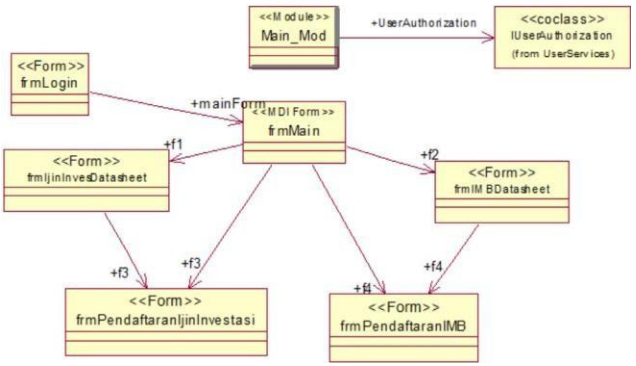
Gambar 4. 1 Object Formulir



Pengujian Class

Menguji terhadap semua operation yg ada dan perubahan atribut-atributnya.

Gambaran UI Sistem Pendaftaran Perijinan

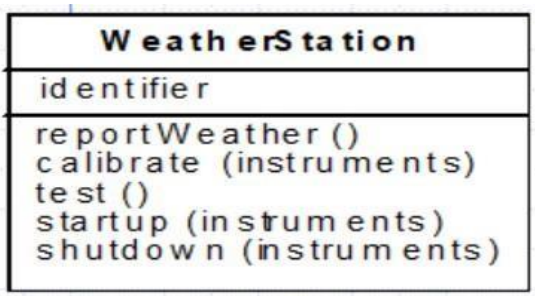


Gambar 4. 2 Pengujian Sistem

Cluster Testing

Cluster testing digunakan untuk test integrasi terhadap kooperatif object. Identifikasi clusters menggunakan knowledge operation objects dan system features yang diimplementasikan oleh cluster tersebut.

Object class testing

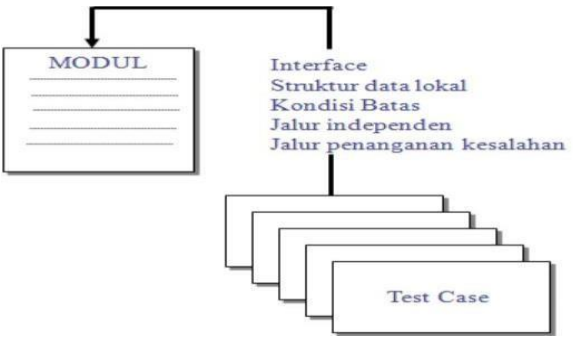


Gambar 4. 3 Object Class Testing

1. Test cases dibutuhkan untuk semua operations
2. Menggunakan state model untuk mengidentifikasi state transitions testing
3. Contoh testing sequences

Pengujian Unit

- Berfokus pada inti terkecil dari desain perangkat lunak yaitu modul
- Biasanya berorientasi pada white box
- Checklist untuk pengujian interface



Gambar 4. 4 Pengujian Unit

1. Apakah jumlah parameter input sama dengan jumlah argumen?
2. Apakah antara atribut dan parameter argumen sudah cocok?
3. Apakah antara sistem satuan parameter dan argumen sudah cocok?
4. Apakah jumlah argumen yang ditransmisikan ke modul yang dipanggil sama dengan atribut parameter?
5. Apakah atribut dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan atribut parameter?
6. Apakah sistem unit dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan sistem satuan parameter?
7. Apakah jumlah atribut dan urutan argumen ke fungsi-fungsi

built-in sudah benar?

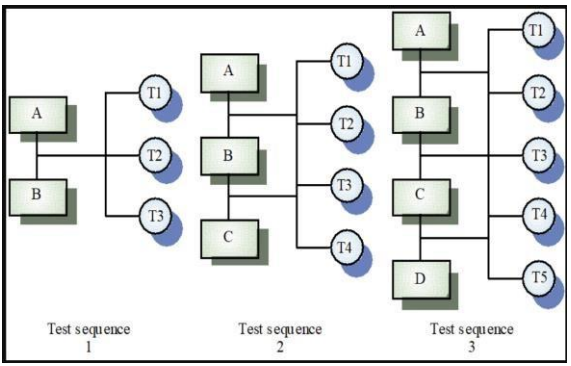
8. Adakah referensi ke parameter yang tidak sesuai dengan poin entri yang ada?
9. Apakah argumen input only diubah?

Seberapa baik sistem yang sudah dibangun ?

- IDua Aspek yang dipertimbangkan:
  1. Apakah implementasi sudah sesuai dengan spesifikasi ?
  2. Apakah spesifikasi sesuai dengan kebutuhan user ?
- Ivalidasi
  1. Apakah sistem yang dikembangkan sudah benar?"
  2. Pengujian dimana sistem ketika diimplementasikan sesuai dengan yang iharapkan
- IVerifikasi
  1. Apakah sistem dikembangkan dengan cara yang benar ?"
  2. Pengujian apakah sistem sudah sesuai dengan spesifikasi

Integration testing

- Pengujian keseluruhan system atau sub-system yang terdiri dr komponen yg terintegrasi.
- Test integrasi menggunakan black-box dengan test case ditentukan dari spesifikasi.
- Kesulitannya adalah menemukan/melokasikan
- Penggunaan Incremental integration testing dapat mengurangi masalah tersebut.



Gambar 4. 5 Intergration Testing

- Top-down testing
  1. Berawal dari level-atas system dan terintegrasi dengan mengganti masing-masing komponen secara top-down dengan suatu stub (program pendek yg mengenerate input ke sub-system yg diuji).
- Bottom-up testing
  1. Integrasi components di level hingga sistem lengkap sudah teruji.

Pendekatan Testing

- Architectural validation
  1. Top-down integration testing lebih baik digunakan dalam menemukan error dalam sistem arsitektur.
- System demonstration
  1. Top-down integration testing hanya membatasi pengujian pada awal tahap pengembangan system.
- Test implementation
  1. Seringkali lebih mudah dengan menggunakan bottom-up integration testing

Interface testing

- Dilakukan kalau module-module dan sub-system terintegrasi dan membentuk sistem yang lebih besar
- Tujuannya untuk medeteksi fault terhadap kesalahan interface atau asumsi yg tidak valid terntang interface tsb.
- Sangat penting untuk pengujian terhadap pengembangan sistem dgn menggunakan pendekatan object-oriented yg didefinisikan oleh object-objectnya.

Kesimpulan

Object-oriented testing

1. Komponen yang diuji adalah class-object.
2. Lebih besar dibandingkan pengujian suatu function sehingga pendekatan white-box testing perlu diperluas.
3. Tidak jelasnya 'top' suatu system untuk top-down integration dan testing.

Testing levels

1. Testing operations pada objects
2. Testing object classes
3. Testing clusters cooperating objects
4. Testing OO system secara lengkap

Pengujian Unit

- Berfokus pada inti terkecil dari desain perangkat lunak yaitu modul
- Biasanya berorientasi pada white box
- Checklist untuk pengujian interface

1. Apakah jumlah parameter input sama dengan jumlah argumen?

2. Apakah antara atribut dan parameter argumen sudah cocok?
3. Apakah antara sistem satuan parameter dan argumen sudah cocok?
4. Apakah jumlah argumen yang ditransmisikan ke modul yang dipanggil sama dengan atribut parameter?
5. Apakah atribut dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan atribut parameter?
6. Apakah sistem unit dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan sistem satuan parameter?
7. Apakah jumlah atribut dan urutan argumen ke fungsi-fungsi built-in sudah benar?
8. Adakah referensi ke parameter yang tidak sesuai dengan poin entri yang ada?
9. Apakah argumen input only diubah?

Seberapa baik sistem yang sudah dibangun ?

- **lDua Aspek yang dipertimbangkan**:

1. Apakah implementasi sudah sesuai dengan spesifikasi ?
2. Apakah spesifikasi sesuai dengan kebutuhan user ?

- **lValidasi**

1. Apakah sistem yang dikembangkan sudah benar?”
2. Pengujian dimana sistem ketika diimplementasikan sesuai dengan yang iharapkan

- **lVerifikasi**

1. Apakah sistem dikembangkan dengan cara yang benar ?”
2. Pengujian apakah sistem sudah sesuai dengan spesifikasi

Integration testing

1. Pengujian keseluruhan system atau sub-system yang terdiri dr komponen yg terintegrasi.

2. Test integrasi menggunakan black-box dengan test case ditentukan dari spesifikasi.
3. Kesulitannya adalah menemukan/melokasikannya.
4. Penggunaan Incremental integration testing dapat mengurangi masalah tersebut.

Pendekatan Testing

Architectural validation

Top-down integration testing lebih baik digunakan dalam menemukan error dalam sistemarsitektur.

System demonstration

Top-down integration testing hanya membatasi pengujian pada awal tahap pengembangansystem.

Test implementation

Seringkali lebih mudah dengan menggunakan bottom-up integration testing

KEHANDALAN PERANGKATLUNAK

Faktor-Faktor Penyebab Kegagalan Proyek Perangkat Lunak

Hampir semua proyek perangkat lunak (PL) dapat dikatakan mengalami kegagalan, walaupun hanya di bagian tertentu saja. Hal ini dikarenakan, hanya sedikit saja proyek yang dapat memenuhi semua target yang ditetapkan baik dari sisi harga, jadwal waktu penyelesaian, kualitas maupun kebutuhan yang diinginkan pengguna (user).

Hanya saja yang masih menyisakan pertanyaan, mengapa kebanyakan proyek yang gagal tidak pernah dikaji lebih lanjut, bahkan oleh lembaga yang mengalaminya sendiri ? Seandainya hal tersebut dilakukan mungkin akan dianggap sebagai suatu petualangan yang tidak akan membuahkan hasil (pemborosan), sehingga sedikit yang mau menginvestasikan waktu dan uangnya untuk mengumpulkan data dan menganalisanya. Bahkan data yang telah berhasil dikumpulkannya mungkin juga hanya berhasil disusun saja, kemudian disembunyikan untuk melindungi karir dan reputasi lembaga tersebut.

Alhasil pemilik informasi tentang kegagalan- kegagalan proyek sering kali berat untuk mempublikasikannya walau untuk tujuan tertentu (yang baik). Sekali lagi, yang dimaksud dengan kegagalan disini adalah proyek perangkat lunak yang mengalami pembengkakan harga, penyelesaiannya molor atau kualitasnya bermasalah (challenged project) maupun yang mengalami pembatalan secara tiba-tiba (failed project). Dan uraian berikut ini akan menjelaskan beberapa penyebab kegagalan yang sering

terjadi dalam pengembangan proyek perangkat lunak.

Poor User Input

Masukan penting yang diperlukan untuk membangun perangkat lunak (PL) adalah bagaimana sistem yang dibuat tersebut akan dipakai di lapangan. Sedangkan yang berkompeten memberikan masukan dalam hal ini adalah klien (*user*) itu sendiri. Hanya saja yang sering terjadi , bahwa *user* kurang terbuka dalam menyatakan kebutuhannya, padahal hal itu sangat berdampak pada pekerjaan *software developer*. Akibatnya, pihak *developer* harus bekerja keras untuk memahami apa saja yang menjadi bisnis *user*, sehingga bisa memprediksi fasilitas apa saja yang diperlukan. Hal tersebut sering berkelanjutan hingga proses *development* selesai.

Stakeholder Conflichts

Stakeholder bekerja di bawah ilusi bahwa semua orang harus mendapatkan apa yang mereka inginkan. Akibatnya sejumlah alternatif yang berbeda dipersiapkan untuk mengakomodasi semua permintaan yang mungkin timbul. Perbedaan tersebut selanjutnya diekspos oleh developer dan mengakibatkan sistem menjadi ambigu. Konflik dalam stakeholder ini bisa menjadi biang utama kegagalan sebuah proyek, karena pihak developer tidak dapat memahami apa yang sesungguhnya diinginkan oleh stakeholder.

Vague Requirement

Perlu bekerja keras untuk mempelajari apa yang akan terjadi ketika sebuah proyek mulai dikerjakan pada saat kebutuhan yang diinginkan belum jelas. Bisa jadi untuk setiap step yang diambil

kemudian perlu mundur tiga langkah ke belakang. Biaya proyek dan kualitas hasil yang berubah cepat bisa lepas dari kontrol, sehingga perusahaan bisa dipersalahkan bahkan bisa kehilangan kontrak untuk menyelesaikan proyek tersebut.Seperti yang sering terjadi pada sejumlah proyek gagal, bahwa lingkup permasalahan yang tidak cukup sempit tidak bisa memberi peluang yang masuk akal untuk sukses. Oleh karenanya perlu memastikan kebutuhan yang diinginkan user sebelum pekerjaan tersebut dimulai. Kendati demikian, kenyataannya perkembangan kebutuhan masih saja terjadi. Permasalahan ini bisa diatasi bila arsitektur dan prosesnya dibuat mampu mengakomodasi perubahan atau setidaknya dibuat ketentuan jelas bagaimana dan kapan requirement bisa ditambahkan, dilepas dan diimplementasikan, termasuk siapa yang akan menanggung biaya perubahan tersebut.

Poor Cost and Schedule Estimation

Sebenarnya tidak fair menuduh setiap proyek PL pasti mengalami kegagalan, jika tidak mengaitkan antara biaya yang dikeluarkan dengan tujuan yang diinginkan. Sama seperti yang lain, setiap proyek PL juga memiliki jadwal dan biaya minimal yang dapat dicapai. Permasalahan bisa muncul kapan saja manakala orang yang membuat jadwal tidak mendengarkan masukan orang yang biasa mengerjakannya. Misalnya untuk menyelesaikan sebuah program biasanya diperlukan 5 orang selama setahun, jelas berbeda seandainya hanya dikerjakan 4 orang selama 8 bulan, baik dilihat dari sisi desain maupun quality-check-nya.

Skills That Do Not Match The Job

Pengerjaan proyek dengan teknologi tinggi memerlukan seorang

manajer dengan kemampuan teknik yang hebat. Penanggung jawab proyek harus bisa menempatkan orang yang memahami dampak dari resiko teknik yang diambil. Namun demikian, seorang teknolog yang baik tidak perlu diseimbangkan dengan manager yang baik, karena kemampuan manajemen dan pemrograman tidak bisa dipertukarkan.Pada proyek yang lebih besar membutuhkan orang-orang yang cakap dalam planning, oversight, organization dan communication skill; sementara seorang teknolog yang brilian tidak perlu memiliki keahlian tersebut.Sebuah team yang beranggotakan orang-orang dengan gaji lebih besar namun memiliki kemampuan spesialisasi yang baik, akan lebih pantas dipilih daripada sebuah kelompok yang terdiri dari orang-orang dengan gaji rendah yang masih memerlukan berminggu-minggu bahkan berbulan-bulan untuk mempelajari sebuah proses atau teknologi baru sebelum mereka mulai mengerjakannya.

Hidden Cost of Going "Lean andMean"

Sejumlah kegagalan akan dipandang sebagai hasil langsung dari kinerja yang buruk, meskipun sesungguhnya kinerja yang buruk tidak menunjuk pada faktor yang signifikan dalam kegagalan pada kebanyakan proyek. Kegagalan sebuah proyek tidak bisa terlepas dari rencana tujuan yang ingin dicapai. Hal lain yang juga pantas dicermati adanya perbedaan kecenderungan yang terjadi dalam sebuah organisasi. Pada level yang lebih rendah, seorang developer akan sibuk menghitung pengeluaran mereka yang mahal, hasil kerja pegawai, *update software* dan pekerjaan-pekerjaan lainnya, sedangkan pada tingkatan yang lebih tinggi hal tersebut akan dikerjakan oleh seorang spesialis yang memadai. Kebanyakan *software developer* akan menghabiskan setengah dari jam kerja mereka untuk berlama-lama dalam mengerjakan tugas

yang diberikan dengan tanpa melakukan apapun terhadap pekerjaan tersebut. Orang-orang demikian ini sangat tidak berkemampuan dan menjadi isu produktifitas yang sangat serius.

Kesimpulan

Hampir semua proyek perangkat lunak (PL) dapat dikatakan mengalami kegagalan, walaupun hanya di bagian tertentu saja. Hal ini dikarenakan, hanya sedikit saja proyek yang dapat memenuhi

Kata Pengantar

Jaminan kualitas perangkat lunak (Software Quality Assurance / SQA) adalah aktifitas pelindung yang diaplikasikan pada seluruh perangkat lunak. SQA Meliputi :

1. Pendekatan Manajemen Kualitas 14
2. Teknologi rekayasa perangkat lunak yang efektif (metode dan piranti)
3. Kajian teknik formal yang diaplikasikan pada keseluruhan proses perangkat lunak
4. Strategi pengujian multitiered (deret bertingkat)
5. Kontrol dokumentasi perangkat lunak dan perubahan yang dibuat untuknya.
6. Prosedur untuk menjamin kesesuain dengan standar pengembangan perangkat lunak (bila dapat diaplikasikan)
7. Mekanisme pengukuran dan pelaporan

Biaya Kualitas dapat di bagi ke dalam biaya biaya yang dihubungkan dengan pencegahan,penilaian dan kegagalan.
Biaya pencegahan meliputi :

- Perencanaan kualitas
- Kajian teknis formal
- Perlengkapan pengujian

- Pelatihan

Biaya Penilaian meliputi aktivitas untuk memperoleh wawasan mengenai kondisi produk “pertamakali” pada masing masing proses. Contoh biaya penilaian meliputi :

- Inspeksi in-Proses dan interproses
- Pemeliharaan dan kalibrasi peralatan
- Pengujian

Biaya Kegagalan adalah biaya yang akan hilang bila tidak ada cacat yang muncul sebelum produkdisampaikan kepada pelanggan. Biaya kegagalan dapat dibagilagi ke dalam biaya :

Biaya Kegagalan Internal

Adalah biaya yang diadakan bila kita mendeteksi suatu kesalahan dalam produk sebelum produk dipasarkan. Biaya kegagalan internal meliputi :

- Pengerjaan kembali
- Perbaikan
- Analisis Mode Kegagalan

Biaya Kegagalan Eksternal

Adalah biaya yang berhubungan dengan cacat yang ditemukan setelah produk disampaikan kepada pelanggan. Biaya Kegagalan Eksternal meliputi :

- Resolusi Keluhan
- Penggantian dan Pengembalian Produk
- Dukungan help Line
- Kerja Jaminan

Jaminan Kualitas Perangkat Lunak

Kualitas perangkat lunak dapat didefinisikan sebagai : Konformasi (Tingkat kesesuaian suatu produk terhadap hal yang telah ditetapkan) terhadap kebutuhan fungsional dan kinerja yang

dinyatakan secara eksplisit, standarperkembangan yang didokumentasikan secara eksplisit, dan karakteristik implisit yang diharapkan bagi semua perangkat lunak yang dikembangkan secara profesional.

Definisi tersebut berfungsi untuk menekankan tiga hal penting, yaitu :

Kebutuhan perangkat lunak merupakan pondasi yang melaluinya Kualitas diukur. Kurangnya penyesuaian terhadap kebutuhan juga menunjukkan rendahnya kualitas.

Standar yang telah ditentukan menetapkan serangkaian kriteria pengembangan yang menuntun cara perangkat lunak direkayasa. Jika kriteria tersebut tidak diikuti, hampir pasti menimbulkan kualitas yang kurang baik.

Ada serangkaian kebutuhan implisit yang sering tidak dicantumkan (misalnya kebutuhan akan kemampuan pemeliharaan yang baik). Bila perangkat lunak dapat berhasil menyesuaikan dengan kebutuhan eksplisitnya, tetapi gagal memenuhi kebutuhan implisitnya, maka kualitas perangkat lunak tersebut perlu diragukan.

Aktifitas SQA

Jaminan Kualitas perangkat lunak terdiri dari berbagai tugas yang berhubungan dengan dua konstituen yang berbeda, perekayasa perangkat lunak yang mengerjakan kerja teknis dan kelompok SQA yang bertanggung jawab terhadap perencanaan jaminan kualitas, kesalahan, penyimpanan rekaman, analisis dan pelaporan.

Tugas Kelompok SQA adalah membantu tim rekayasa perangkat lunak dalam pencapaian produk akhir yang berkualitas tinggi. The Software Engineering Institute merekomendasikan serangkaian aktifitas SQA yang menekankan rencana jaminan kualitas, kesalahan, penyimpanan rekaman, analisis dan pelaporan.

Berikut ini akatifitas yang dilakukan (difasilitasi) oleh kelompok SQA yang independen Menyiapkan Rencana SQA untuk Suatu Proyek berpartisipasi dalam pengembangan deskripsi proses pengembangan proyek. Tim rekayasa perangkat lunak memilih sebuah proses bagi kerja yang akan dilakukan. Mengkaji aktifitas rekayasa perangkat lunak untuk memverifikasi pemenuhan proses perangkat lunak yang sudah ditentukan.

Kelompok SQA mengidentifikasi, mendokumentasi, dan menelusuri deviasi proses dan membuktikan apakah koreksi sudah dilakukan. Mengaudit produk kerja perangkat lunak yang ditentukan untuk membuktikan kesesuaian dengan produk kerja yang ditentukan tersebut sebagai bagaian dari proses perangkat lunak. Memastikan bahwa deviasi pada kerja dan produk kerja perangkat lunak didokumentasi dan ditangani sesuai prosedur pendokumentasian. Mencatat ketidaksesuaian dan melaporkannya kepada manajemen senior. Item item yang tidak sesuai ditelursuri sampai item itu diubah.

Kajian Perangkat Lunak

Kajian perangkat lunak adalah suatu “filter” bagi proses rekayasa perangkat lunak, yaitu kajian yang diterapkan pada berbagai titik selama pengembangan perangkat lunak dan berfungsi untuk mencari kesalahan yang kemudian akan dihilangkan. Kajian perangkat lunak berfungsi untuk “memurnikan” produk kerja perangkat lunak yang terjadi sebagai hasil dari analisis, desain dan pengkodean.

Kajian teknik Formal

Kajian Teknik Formal (Formal Technique Research) adalah aktifitas jaminan kualitas perangkat lunak yang dilakukan oleh perekayasa

perangkat lunak.

Tujuan FTR adalah :

Menemukan kesalahan dalam fungsi, logika atau implementasinya dalam berbagai representasi perangkat lunak

Membuktikan bahwa perangkat lunak disajikan sesuai dengan standar yang sudah ditentukan sebelumnya.

Mencapai perangkat lunak yang dikembangkan dengan cara seragam.

Membuat proyek lebih dapat dikelola.

Sebagai tambahan, FTR berfungsi sebagai dasar pelatihan yang memungkinkan perekayasa junior mengamati berbagai pendekatan yang berbeda terhadap analisis perangkat lunak, desain dan implementasi. FTR juga berfungsi untuk mengembangkan backup dan kontinuitas karena sejumlah orang mengenal baik bagian-bagian perangkat lunak yang tidak mereka ketahui sebelumnya.

Jaminan Kualitas

Jaminan kualitas terdiri atas fungsi auditing dan pelaporan manajemen. Tujuan jaminan kualitas untuk memberikan data yang diperlukan oleh manajemen untuk menginformasikan masalah kualitas produk, sehingga dapat memberikan kepastian dan konfidensi bahwa kualitas produk dapat memenuhi sasaran.

Biaya Kualitas

Biaya kualitas menyangkut semua biaya yang diadakan untuk mengejar kualitas atau untuk menampilkan kualitas yang berhubungan dengan aktifitas. Studi tentang biaya kualitas dilakukan untuk memberikan garis dasar bagi biaya yang sedang digunakan, untuk mengidentifikasi kemungkinan pengurangan biaya serta memberikan basis perbandingan yang ternormalisasi.

Reliabilitas Perangkat Lunak

Tidak diragukan lagi bahwa reliabilitas sebuah program komputer merupakan suatu elemen yang penting. Bila sebuah program berkali-kali gagal untuk melakukan kinerja, maka sedikit meragukan apakah faktor kualitas perangkat lunak yang lain dapat diterima. Reliabilitas perangkat lunak, tidak seperti faktor kualitas yang lain, dapat diukur, diarahkan, dan diestimasi dengan menggunakan data pengembangan historis. Reliabilitas perangkat lunak didefinisikan dalam bentuk statistik sebagai " Kemungkinan operasi program komputer bebas kegagalan di dalam suatu lingkungan tertentu dan waktu tertentu".

Contoh :
Program X diperkirakan memiliki reliabilitas 0.96 pada delapan jam pemrosesan yang dilalui. Dengan kata lain, jika program X akan dieksekusi 100 kali dan membutuhkan delapan jam waktu pemrosesan yang dilalui (waktu eksekusi), dia akan beroperasi dengan benar (tanpa kegagalan) 96 kali dari 100 kali pelaksanaan.
Keamanan Perangkat Lunak dan Analisis Resiko Keamanan Perangkat lunak dan analisis resiko adalah aktifitas jaminan kualitas perangkat lunak yang berfokus pada identifikasi dan penilaian resiko potensial yang mungkin berpengaruh negatif terhadap perangkat lunak dan menyebabkan seluruh sistem menjadi gagal. Jika resiko dapat diidentifikasi pada awal proses rekayasa perangkat lunak, maka ciri-ciri desain perangkat lunak dapat ditetapkan sehingga akan mengeliminasi atau mengontrol resiko potensial.

Standar Kualitas ISO

Sistem jaminan kualitas dapat didefinisikan sebagai struktur, tanggung jawab, prosedur, proses dan sumber sumber daya

organisasi untuk mengimplementasi manajeman kualitas.

ISO 9000 menjelaskan elemen jaminan kualitas dalam bentuk yang umum yang dapat diaplikasikan pada berbagai bisnis tanpa memandang produk dan jasa yang ditawarkan. Elemen elemen tersebut mencakup struktur, prosedur, proses, organbisasi dan sumber daya yang dibutuhkan untuk mengimplementasi rencana kualitas , kualitas kontrol, jaminan kualitas dan pengembangan kualitas.

Agar terdaftar dalam satu model sistem jaminan kualitas yang ada pada ISO 9000, sistem kualitas dan operasi perusahaan diperiksa oleh auditor untuk memeriksa kesesuaiannya dengan standar dan operasi efektif. Bila registrasi itu berhasil, perusahaan diberi sertifikasi dari bada registrasi yang diwakili oleh auditor. Audit pengawasan tengah tahunan terus dilakukan untuk memastikan kesesuaiannya dengan standar yang sudah ditetapkan.

STANDAR ISO 9001

ISO 9001 adalah standar jaminan kualitas yang berlaku untuk rekayasa perangkat lunak. Standar tersebut berisi 20 syarat yang harus ada untuk mencapai sistem jaminan kualitas yang efektif , yaitu:

1. Tanggung jawab manajeman
2. Sistem Kualitas
3. Kajian Kontrak
4. Kontrol Desain
5. Kontrol data dan dokumen
6. Pembelian
7. Kontrol terhadap produk yang disuplai oleh pelanggan
8. Identifikasi dan kemampuan penelusuran produk
9. Kontrol Proses
10. Pemeriksaan dan pengujian
11. Kontrol Pemeriksaan, pengukuran dan perlengkapan pengujian

12. Pemeriksaan dan status pengujian
13. Kontrol ketidaksesuain produk
14. Tindakan preventif dan korektif
15. Penanganan, penyimpanan, pengepakan, preservasi dan penyampaian.
16. Kontrol terhadap catatan kualitas
17. Audit kualitas internal
18. Pelatihan
19. Pelayanan
20. Teknik statistik

Untuk dapat didaftar dalam ISO 9001, organisasi perangkat lunak harus membuat kebijakan dan prosedur yang memberi tekanan pada masing masing syarat tersebut dan kemudian dapat menunjukkan bahwa prosedur dan fungsi itu telah diikuti.

Kesimpulan

Konsep Kualitas

1. Kualitas

American Heritage Dictionary mendefinisikan kata kualitas sebagai "Sebuah Karakteristik atau atribut dari sesuatu". Sebagai atribut dari sesuatu, kualitas mengacu pada karakteristik yang dapat diukur, sesuatu yang dapat kita bandingkan dengan standar yang sudah diketahui, seperti panjang, warna, sifat kelistrikan, kelunakan dsb. Tetapi perangkat lunak, yang sebagian besar merupakan entitas intelektual, lebih menantang untuk dikarakterisasi daripada objek fisik.

2. Kontrol Kualiatas

Kontrol kualitas merupakan serangkaian pemeriksaan, kajian dan pengujian yang digunakan pada keseluruhan siklus pengembangan untuk memastikan bahwa setiap produk memenuhi persyaratan yang ditetapkan. Kontrol kualitas mencakup loop (kalang) umpan balik pada proses yang menciptakan produk kerja. Kombinasi

pengukuran dan umpan balik memungkinkan kita memperbaiki proses bila produk kerja yang diciptakan gagal memenuhi spesifikasi mereka. Pendekatan tersebut memandang kontrol kualitas sebagai bagian dari proses pemanufakturan

3. Jaminan Kualitas

Jaminan kualitas terdiri atas fungsi auditing dan pelaporan manajemen. Tujuan jaminan kualitas untuk memberikan data yang diperlukan oleh manajemen untuk menginformasikan masalah kualitas produk, sehingga dapat memberikan kepastian dan konfidensi bahwa kualitas produk dapat memenuhi sasaran.

4. Biaya Kualitas

Biaya kualitas menyangkut semua biaya yang diadakan untuk mengejar kualitas atau untuk menampilkan kualitas yang berhubungan dengan aktifitas. Studi tentang biaya kualitas dilakukan untuk memberikan garis dasar bagi biaya yang sedang digunakan, untuk mengidentifikasi kemungkinan pengurangan biaya serta memberikan basis perbandingan yang ternormalisasi.

5. Jaminan Kualitas Perangkat Lunak

Kualitas perangkat lunak dapat didefinisikan sebagai : Konformasi (Tingkat kesesuaian suatu produk terhadap hal yang telah ditetapkan) terhadap kebutuhan fungsional dan kinerja yang dinyatakan secara eksplisit, standarperkembangan yang didokumentasikan secara eksplisit, dan karakteristik implisit yang diharapkan bagi semua perangkat lunak yang dikembangkan secara profesional.

6. Kajian Perangkat Lunak

Kajian perangkat lunak adalah suatu “filter” bagi proses rekayasa perangkat lunak, yaitu kajian yang diterapkan pada berbagai titik selama pengembangan perangkat lunak dan berfungsi untuk mencari kesalahan yang kemudian akan dihilangkan. Kajian perangkat lunak berfungsi untuk “memurnikan” produk kerja perangkat lunak yang terjadi sebagai hasil dari analisis, desain dan pengkodean

7. Reliabilitas Perangkat Lunak

Tidak diragukan lagi bahwa reliabilitas sebuah program komputer

merupakan suatu elemen yang penting. Bila sebuah program berkali-kali gagal untuk melakukan kinerja, maka sedikit meragukan apakah faktor kualitas perangkat lunak yang lain dapat diterima.

TECHNICAL METRICS PERANGKAT LUNAK

Kriteria Penjaminan Kualitas PerangkatLunak

Dalam salah satu referensi disebutkan bahwa yang dimaksud dengan software quality adalah pemenuhan terhadap kebutuhan fungsional dan kinerja yang didokumentasikan secara eksplisit, pengembangan standar yang didokumentasikan secara eksplisit, dan sifat-sifat implisit yang diharapkan dari sebuah software yang dibangun secara profesional (Dunn, 1990). Berdasarkan definisi di atas terlihat bahwa sebuah software dikatakan berkualitas apabila memenuhi tiga ketentuan pokok :

1. Memenuhi kebutuhan pemakai – yang berarti bahwa jika software tidak dapat memenuhi kebutuhan pengguna software tersebut, maka yang bersangkutan dikatakan tidak atau kurang memiliki kualitas
2. Memenuhi standar pengembangan software – yang berarti bahwa jika cara pengembangan software tidak mengikuti metodologi standar, maka hampir dapat dipastikan bahwa kualitas yang baik akan sulit atau tidak tercapai.
3. Memenuhi sejumlah kriteria implisit – yang berarti bahwa jika salah satu kriteria implisit tersebut tidak dapat dipenuhi, maka software yang bersangkutan tidak dapat dikatakan memiliki kualitas yang baik.

McCall dan kawan-kawan pada tahun 1977 telah mengusulkan suatu penggolongan faktor-faktor atau kriteria yang mempengaruhi kualitas software. Pada dasarnya, McCall

menitikberatkan faktor-faktor tersebut menjadi tiga aspek penting, yaitu yang berhubungan dengan :

a. Sifat-sifat operasional dari software (Product Operations).
b. Kemampuan software dalam menjalani perubahan (Product Revision)
c. Daya adaptasi atau penyesuaian software terhadap lingkungan baru (Product Transition)

Product Operations

Sifat-sifat operasional suatu software berkaitan dengan hal-hal yang harus diperhatikan oleh para perancang dan pengembang yang secara teknis melakukan penciptaan sebuah aplikasi. Hal-hal yang diukur di sini adalah yang berhubungan dengan teknis analisa, perancangan, dan konstruksi sebuah software. Faktor faktor McCall yang berkaitan dengan sifat-sifat operasional software adalah:

a. Correctness – sejauh mana suatu software memenuhi spesifikasi dan mission objective dari users.
b. Reliability – sejauh mana suatu software dapat diharapkan untuk melaksanakan fungsinya dengan ketelitian yang diperlukan.
c. Efficiency – banyaknya sumber daya komputasi dan kode program yang dibutuhkan suatu software untuk melakukan fungsinya
d. Integrity – sejauh mana akses ke software dan data oleh pihak yang tidak berhak dapat dikendalikan.
e. Usability – usaha yang diperlukan untuk mempelajari, mengoperasikan, menyiapkan input, dan mengartikan output dari software.

Product Revision

Setelah sebuah software berhasil dikembangkan dan diimplementasikan, akan terdapat berbagai hal yang perlu diperbaiki berdasarkan hasil uji coba maupun evaluasi. Sebuah software yang dirancang dan dikembangkan dengan baik, akan dengan mudah dapat direvisi jika diperlukan.

Seberapa jauh software tersebut dapat diperbaiki merupakan 23 Faktor lain yang harus diperhatikan. Faktor-faktor McCall yang berkaitan dengan kemampuan software untuk menjalani perubahan adalah:

a. Maintainability – usaha yang diperlukan untuk menemukan dan memperbaiki kesalahan (error) dalam software.
b. Flexibility – usaha yang diperlukan untuk melakukan modifikasi terhadap software yang operasional.
c. Testability – usaha yang diperlukan untuk menguji suatu software untuk memastikan apakah melakukan fungsi yang dikehendaki atau tidak.

Product Transition

Setelah integritas software secara teknis telah diukur dengan menggunakan faktor product operational dan secara implementasi telah disesuaikan dengan faktor product revision, faktor terakhir yang harus diperhatikan adalah faktor transisi – yaitu bagaimana software tersebut dapat dijalankan pada beberapa platform atau kerangka sistem yang beragam.

Faktor-faktor McCall yang berkaitan dengan tingkat adaptibilitas software terhadap lingkungan baru.

a. Portability – usaha yang diperlukan untuk mentransfer software dari suatu hardware dan/atau sistem software tertentu agar dapat berfungsi pada hardware dan/atau sistem software lainnya.

b. Reusability – sejauh mana suatu software (atau bagian software) dapat dipergunakan ulang pada aplikasi lainnya
c. Interoperability – usaha yang diperlukan untuk menghubungkan satu software dengan lainnya
d. Integrity – sejauh mana akses ke software dan data oleh pihak yang tidak berhak dapat dikendalikan.
e. Usability – usaha yang diperlukan untuk mempelajari, mengoperasikan, menyiapkan input, dan mengartikan output dari software.

Product Revision

Setelah sebuah software berhasil dikembangkan dan diimplementasikan, akan terdapat berbagai hal yang perlu diperbaiki berdasarkan hasil uji coba maupun evaluasi. Sebuah software yang dirancang dan dikembangkan dengan baik, akan dengan mudah dapat direvisi jika diperlukan.
Seberapa jauh software tersebut dapat diperbaiki merupakan 23

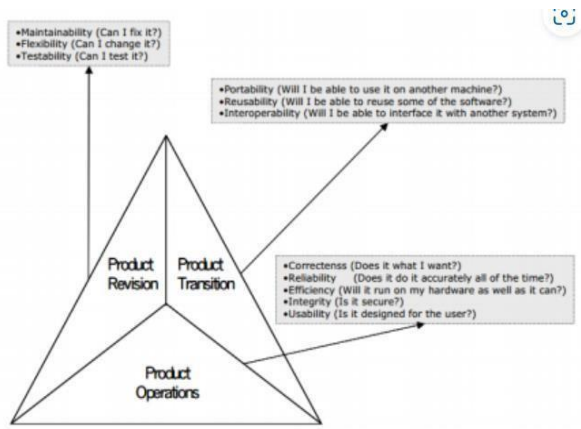
Faktor lain yang harus diperhatikan. Faktor-faktor McCall yang berkaitan dengan kemampuan software untuk menjalani perubahan adalah:

a. Maintainability – usaha yang diperlukan untuk menemukan dan memperbaiki kesalahan (error) dalam software.
b. Flexibility – usaha yang diperlukan untuk melakukan modifikasi terhadap software yang operasional.
c. Testability – usaha yang diperlukan untuk menguji suatu software untuk memastikan apakah melakukan fungsi yang dikehendaki atau tidak.

Product Transition

Setelah integritas software secara teknis telah diukur dengan menggunakan faktor product operational dan secara implementasi telah disesuaikan dengan faktor product revision, faktor terakhir yang harus diperhatikan adalah faktor transisi – yaitu bagaimana software tersebut dapat dijalankan pada beberapa platform atau kerangka sistem yang beragam.
Faktor-faktor McCall yang berkaitan dengan tingkat adaptibilitas software terhadap lingkungan baru.

a. Portability – usaha yang diperlukan untuk mentransfer software dari suatu hardware dan/atau sistem software tertentu agar dapat berfungsi pada hardware dan/atau sistem software lainnya.
b. Reusability – sejauh mana suatu software (atau bagian software) dapat dipergunakan ulang pada aplikasi lainnya.
c. Interoperability – usaha yang diperlukan untuk menghubungkan satu software dengan lainnya.



Sumber : Pressman, 1994

Gambar 5. 1 Product Transition

Teknik Pengukuran

Menimbang tingkat kesulitan yang dihadapi para programmer dalam mengukur secara langsung dan kuantitatif kualitas software yang dikembangkan berdasarkan pembagian yang diajukan McCall di atas, sebuah formula diajukan untuk mengukur faktor-faktor software quality secara tidak langsung menurut hubungan

Adapun metric yang dipakai dalam skema pengukuran di atas adalah sebagai berikut:

a. Auditability – kemudahan untuk memeriksa apakah software memenuhi standard atau tidak.
b. Accuracy – ketelitian dari komputasi dan kontrol.
c. Communication Commonality – sejauh mana interface, protokol, dan bandwidth digunakan.
d. Completeness – sejauh mana implementasi penuh dari fungsi-fungsi yang diperlukan telah tercapai.
e. Conciseness – keringkasan program dalam ukuran LOC (line of commands).
f. Consistency – derajat penggunaan teknik-teknik desain dan dokumentasi yang seragam pada seluruh proyek pengembangan software.
g. Data Commonality – derajat penggunaan tipe dan struktur data baku pada seluruh program.
h. Error Tolerance – kerusakan yang terjadi apabila program mengalami error.
i. Execution Efficiency – kinerja run-time dari program.
j. Expandability – sejauh mana desain prosedur, data, atau arsitektur dapat diperluas.
k. Generality – luasnya kemungkinan aplikasi dari komponen-komponen program.

l. Hardware Independence – sejauh mana software tidak bergantung pada kekhususan dari hardware tempat software itu beroperasi.
m. Instrumentation – sejauh mana program memonitor operasidirinya sendiri dan mengidentifikasi error yang terjadi.
n. Modularity – functional independence dari komponen-komponen program.
o. Operability – kemudahan mengoperasikan program.
p. Security – ketersediaan mekanisme untuk mengontrol dan melindungi program dan data terhadap akses dari pihak yang tidak berhak.
q. Self-Dokumentation – sejauh mana source-code memberikan dokumentasi yang berarti.
r. Simplicity – Kemudahan suatu program untuk dimengerti.
s. Traceability – kemudahan merujuk balik implementasi atau komponen program ke kebutuhan pengguna software.
t. Training – sejauh mana software membantu pemakaian baru untuk menggunakan sistem.

Kesimpulan

Kriteria Penjaminan Kualitas Perangkat Lunak

dimaksud dengan software quality adalah pemenuhan terhadap kebutuhan fungsional dan kinerja yang didokumentasikan secara eksplisit, pengembangan standar yang didokumentasikan secara eksplisit, dan sifat-sifat implisit yang diharapkan dari sebuah software yang dibangun secara profesional (Dunn, 1990).

McCall dan kawan-kawan pada tahun 1977 telah mengusulkan suatu penggolongan faktor-faktor atau kriteria yang mempengaruhi kualitas software. Pada dasarnya, McCall

menitikberatkan faktor-faktor tersebut menjadi tiga aspek penting, yaitu yang berhubungan dengan :

a. Sifat-sifat operasional dari software (Product Operations).
b. Kemampuan software dalam menjalani perubahan (Product Revision)
c. Daya adaptasi atau penyesuaian software terhadap lingkungan baru (Product Transition)

Product Operations

Sifat-sifat operasional suatu software berkaitan dengan hal-hal yang harus diperhatikan oleh para perancang dan pengembang yang secara teknis melakukan penciptaan sebuah aplikasi. Hal-hal yang diukur di sini adalah yang berhubungan dengan teknis analisa, perancangan, dan konstruksi sebuah software.
faktor lain yang harus diperhatikan. Faktor-faktor McCall yang berkaitan dengan kemampuan software untuk menjalani perubahan adalah:

1. Maintainability – usaha yang diperlukan untuk menemukan dan memperbaiki kesalahan (error) dalam software.
2. Flexibility – usaha yang diperlukan untuk melakukan modifikasi terhadap software yang operasional.
3. Testability – usaha yang diperlukan untuk menguji suatu software untuk memastikan apakah melakukan fungsi yang dikehendaki atau tidak.

Faktor-faktor McCall yang berkaitan dengan tingkat adaptibilitas software terhadap lingkungan baru.

1. Portability – usaha yang diperlukan untuk mentransfer software dari suatu hardware dan/atau sistem software tertentu agar dapat berfungsi pada hardware dan/atau sistem software lainnya.

2. Reusability – sejauh mana suatu software (atau bagian software) dapat dipergunakan ulangpada aplikasi lainnya.
3. Interoperability – usaha yang diperlukan untuk menghubungkan satu software dengan lainnya.

Pengujian Perangkat Lunakpada arsitektur client/server

Pada saat perangkat lunak komputer menjadi semakin kompleks,maka kebutuhan akan pendekatan pengujian yang khusus juga makin berkembanga. Metode pengujian black-box dan white box yang dibicarakan sebelumnya dapat diaplikasikan pada semua lingkungan, arsitektur dan aplikasi, tetapi kadang-kadang dalam pengujian diperlukan pedoman dan pendekatan yang unik. Pada bagian ini akan dibahas pedoman pengujian bagi lingkungan arsitektur dan aplikasi khusus yang umumnya ditemui oleh para perekayasa perangkat lunak.

Pengujian GUI

Graphical User Interfaces (GUI) menyajikan tantangan yang menarik bagi para perekayasa. Karena komponen reusable berfungsi sebagai bagian dari lingkungan pengembangan GUI, pembuatan interface pemakai telah menjadi hemat waktu dan lebi teliti. Pada saat yang sama, Kompleksitas GUI telah berkembanga, menimbulkan kesulitan yang lebih besar di dalam desain dan eksekusi test case.

Karena GUI modern memiliki bentuk dan cita rasa yang sama maka dapat dilakukan sederetan pengujian standar. Pertanyaan berikut dapat berfungsi sebagai panduan untuk serangkaian pengujian generik untuk GUI :

Untuk windows :

Apakah window akan membuka secara tepat berdasarkan tipe yang sesuai atau perintah berbasis menu ?

Dapatkah window di-resize, digerakkan atau digulung ?

Apakah window dengan cepat muncul kembali bila dia ditindih dan kemudian dipanggil lagi ?

Apakah semua fungsi yang berhubungan dengan window dapat diperoleh bila diperlukan ?

Apakah semua fungsi yang berhubungan dengan window operasional ?

Apakah semua menu pull down, tool bar,scroll bar, kotak dialog, tombol ikon dan kontrol yang laina dapat diperoleh dan dengan tepat ditampilkan untu window tersebut ?

Padasaat window bertingkat ditampilkan, apakah nama window tersebut direpresentasikan secara tepat

Apakah window yang aktif disorot secara tepat ?

Bila multitasking digunakan, apakah semua window diperbarui pada waktu yang sesuai ?

Apakah pemilihan mouse bertingkat atau tidak benar di dalam window menyebabkan efek samping yang tidak diharapkan ?

Apakah audio prompt dan atau color prompt ada di dala window atau sebagai konsekuensi dari operasi windoe disaajikan menurut spesifikasi ?

Apakah window akan menutup secara tepat ?

Untuk menu pull down dan operasi mouse :

Apakah menu bar yang sesuai ditampilkan di dalam konteks yang sesuai ?

Apakah menu bar aplikasi menampilkan fitur-fitur yang terkait dengan sistem (misal tampilan jam) ?

Apakah operasi menu pull down bekerja secara tepat ?

Apakah menu breakway, palette dan tool bar bekerja secara tepat ?

Apakah semua fungsi menu dan subfungsi pulldown didaftar secara tepat ?

Apakah semua fungsi menu dapat dituju secara tepat oleh mouse ?

Apakah typface, ukuran dan format teks benar ?

Apakah fungsi menu disorot berdasarkan konteks operasi ang sedang berlangsung di dalam suatu window ?
Apakah semua menu function bekerja seperti diiklankan ?
Apakah nama-nama menu funvtion bersifat self explanatory ?

Apakah help dapat diperoleh untuk masing-masing item menu, apakah dia sensitif terhadap konteks ?
Apakah operasi mouse dikenali dnegna baik pada seluruh konteks interaktif ?
Bila klik ganda diperlukan, apakah operasi dikenali di dalam konteks ?
Jika mouse mempunyai tombol ganda, apakah tombol itu dikenali sesuai konteks ?
Apakah kursor, indikator permosesan (seperti jam) dan pointer secara tepat berubah pada saat operasi yang berbeda dipanggil ?

Entri Data :
Apakah entri data alfanumeris dipantulkan dan diinput ke sistem ?
Apakah mode grafik dari entri (misal, slide bar) bekerja dengan baik ?
Apakah data invalid dikenali dengan baik ?
Apakah pesan input data sangat pintar ?

Sebagai tambahan untuk pedoman tersebut, grafik pemodelan keadaan yang terbatas dapat digunakan untuk melakukan sederetan pengujian yang menekankan objek program dan data spesifik yang relevan dengan GUI.
Karena sejumlah besar permutasi yang bekesesuaian dengan operasi GUI, maka pengujian harus didekati denga menggunakan peranti otomatis. Sudah banyak peranti pengujian GUI yang muncul dipasaran selama beberapa tahun terakhir.

Pengujian Arsitektur Client Server

Arsitektur client/server (C/S) menghadirkan tantangan yang berarti bagi para penguji perangkat lunakan. Sifat terdistribusi dari lingkungan client/server, masalah kinerja yang berhubungan dengan pemrosesan transaksi, kehadiran potensial dari sejumlah platform perangkat keras yang berbeda, kompleksitas komunikasi jaringan,kebutuahn aka layanan client multipel dari suatu database terpusat dan persyaratan koordinasi yang dibebabkan pada server, semua secara bersama-sama membuat pengujian terhadap arsitektur C/S dan perangkat lunak yang ada didalamnya menjadi jauh lebih sulit daripada pengujian aplikasi yang berdiri sendiri. Kenyataannya studi industri terakhir menunjukkan pertambahan berarti di dalam waktu pengujian dan biaya ketika lingkungan C/S dikembangkan.

Pengujian Dokumentasi dan FasilitasHelp

Istilah “pengujianperangkat lunak” memunculkan citra terhdapa sejumlah besar test case yang disiapkan untuk menggunakan program komputer dan data yang dimanipulasi oleh program. Dengan melihat kembali definisi perangkat lunak yang disajikan di awal, penting untuk dicatat bahwa pengujian harus berkembang ke elemen ketiga dari konfigurasi perangkat lunak, yaitu “dokumentasi”.

Kesalahan dala dokumentasi dapat menghancurkan penerimaan program seperti halnya kesalahan pada data atu kode sumber. Tidak ada yang lebih membuat frustasi dibanding mengikuti tuntuan pemakai secara tepat dan mendapatkan hasil atau

tingkah laku yang tidak sesuai dengan yang diprediksi oleh dokumen. Karena itulah pengujian dokumentasi harus menjadi suatu bagian yang berarti dari setiap rencana pengujian perangkat lunak.

Pengujian dokumentasi dapat didekati dalam dua fase. Fase pertama, kajian teknis formal yang menguji kejelasan editorial dokumen. Fase kedua, live test, menggunakan dokumentasi dalam kaitannya dengan penggunaan program aktual.

Live test untuk dokumentasi dapat didekati dengan menggunaka teknik yang analog dnegan berbagai metode pengujian black box. Pengujian graph-based ddapat digunakan untuk menggambarkan penggunaan program tersebut; partisi ekivalensi dan analisis nilai b atas dapat digunakan untuk menentukan berbagai kelas input dan interaksi yang sesuai.

Pengujian Sistem Real-Time

Sifat asinkron dan tergantung waktu yang ada pada banyak aplikasi real time menambahkan elemen baru yang sulit dan potensial untuk bauran pengujian-waktu. Tidak hanya desainer teset case yang harus memeprtimbangkan test case balck box dan white box etapi juga penanganan kejadian (yaitu pemrosesan interupsi), timing data dan paralelisme tugas-tugas (proses) yang menangani data. Pada banyak situasi, data pengujian yang diberikan pada saat sebuah sistem real time ada dalam satu keadaan akan menghasilkan pemrosesan yang baik, sementara data yang sama yang diberikan pada saat sistem berada dalam keadaan yang berbeda dapat menyebabkan kesalahan.

Contohnya, perangkat lunak real time yang mengontrol alat foto kopi yang baru menerima interupsi operator (yakni operator mesin menekan kunci kontrol seperti reset) dengan tanpa kesalahan pada saat mesin sedang membuan kopian. Interupsi

operator yang sama ini bila diinputkan pada saat mesin ada dalam keadaan “jammed” akan menyebabkan sebuah kode diagnostik yang menunjukkan lokasi jam yang akan hilang (kesalahan). Hubungan erat perangkat lunak real time dan lingkungan perangkat kerasnya dapat juga menyebabkan pengaruh kegagalan perangkat keras pada pemrosesasn perangkat lunak. Kesalahan semacam itu dapat sangat sulit untuk bersimulasi secara realistis.

Kesimpulan

Pada saat perangkat lunak komputer menjadi semakin kompleks,maka kebutuhan akan pendekatan pengujian yang khusus juga makin berkembanga. Metode pengujian black-box dan white box yang dibicarakan sebelumnya dapat diaplikasikan pada semua lingkungan, arsitektur dan aplikasi, tetapi kadang-kadang dalam pengujian diperlukan pedoman dan pendekatan yang unik. Pada bagian ini akan dibahas pedoman pengujian bagi lingkungan arsitektur dan aplikasi khusus yang umumnya ditemui oleh para perekayasa perangkat lunak.

Pengujian GUI

Graphical User Interfaces (GUI) menyajikan tantangan yang menarik bagi para perekayasa. Karena komponen reusable berfungsi sebagai bagian dari lingkungan pengembangan GUI, pembuatan interface pemakai telah menjadi hemat waktu dan lebi teliti. Pada saat yang sama, kompleksitas GUI telah berkembanga, menimbulkan kesulitan yang lebih besar di dalam desain dan eksekusi test case.

Karena GUI modern memiliki bentuk dan cita rasa yang sama maka dapat dilakukan sederetan pengujian standar. Pertanyaan berikut dapat berfungsi sebagai panduan untuk serangkaian pengujian generik untuk GUI.

Pengujian Arsitektur Client Server

Arsitektur client/server (C/S) menghadirkan tantangan yang berarti bagi para penguji perangkat lunakan Sifat terdistribusi dari lingkungan client/server, masalah kinerja yang berhubungan dengan pemrosesan transaksi, kehadiran potensial dari sejumlah platform perangkat keras yang berbeda, kompleksitas komunikasi jaringan,kebutuahn aka layanan client multipel dari suatu database terpusat dan persyaratan koordinasi yang dibebabkan pada server, semua secara bersama-sama membuat pengujian terhadap arsitektur C/S dan perangkat lunak yang ada didalamnya menjadi jauh lebih sulit daripada pengujian aplikasi yang berdiri sendiri. Kenyataannya studi industri terakhir menunjukkan pertambahan berarti di dalam waktu pengujian dan biaya ketika lingkungan C/S dikembangkan.

Pengujian Dokumentasi dan FasilitasHelp

Istilah “pengujianperangkat lunak” memunculkan citra terhdapa sejumlah besar test case yang disiapkan untuk menggunakan program komputer dan data yang dimanipulasi oleh program. Dengan melihat kembali definisi perangkat lunak yang disajikan di awal, penting untuk dicatat bahwa pengujian harus berkembang ke elemen ketiga dari konfigurasi perangkat lunak, yaitu “dokumentasi”.

Kesalahan dala dokumentasi dapat menghancurkan penerimaan program seperti halnya kesalahan pada data atu kode sumber. Tidak ada yang lebih membuat frustasi dibanding mengikuti

tuntuan pemakai secara tepat dan mendapatkan hasil atau tingkah laku yang tidak sesuai dengan yang diprediksi oleh dokumen. Karena itulah pengujian dokumentasi harus menjadi suatu bagian yang berarti dari setiap rencana pengujian perangkat lunak.

Pengujian dokumentasi dapat didekati dalam dua fase. Fase pertama, kajian teknis formal yang menguji kejelasan editorial dokumen. Fase kedua, live test, menggunakan dokumentasi dalam kaitannya dengan penggunaan program aktual.

Pengujian Sistem Real-Time

Sifat asinkron dan tergantung waktu yang ada pada banyak aplikasi real time menambahkan elemen baru yang sulit dan potensial untuk bauran pengujian-waktu. Tidak hanya desainer teset case yang harus memeprtimbangkan test case balck box dan white box etapi juga penanganan kejadian (yaitu pemrosesan interupsi), timing data dan paralelisme tugas-tugas (proses) yang menangani data. Pada banyak situasi, data pengujian yang diberikan pada saat sebuah sistem real time ada dalam satu keadaan akan menghasilkan pemrosesan yang baik, sementara data yang sama yang diberikan pada saat sistem berada dalam keadaan yang berbeda dapat menyebabkan kesalahan.

PENGUJIAN APLIKASI BERBASIS WEB

Jenis pengujian pada aplikasi web

Pengujian Aplikasi Web adalah serangkaian aktivitas yang berkaitan dengan tujuan : menemukan kesalahan dalam isi, fungsi, kegunaan, kemampuan navigasi, kinerja, kapasitas dan keamanan aplikasi web.

Yang melakukan pengujian web : pengembang web dan stakeholder proyek lainnya (manajer, pelanggan, pengguna akhir).

Secara umum tahapan yang dilakukan ada 7 tahap :

a. Pengujian isi
b. Pengujian antarmuka
c. Pengujian navigasi
d. Pengujian komponen
e. Pengujian konfigurasi
f. Pengujian kinerja
g. Pengujian keamanan

Dimensi Kualitas dalam Pengujian Aplikasi Web

Isi (content), dievaluasi baik di tingkat sintaksis maupun semantik. Di tingkat sintaktis, dokumen-dokumen berbasis teks diuji dalam hal ejaan, tanda baca dan tata bahasa. Di tingkat sematik, aspek yang dinilai adalah kebenaran (informasi yang disajikan), konsistensi (di seluruh objek isi dan objek terkait)dan rendahnya ambiguitas.

Fungsi, diuji untuk menemukan kesalahan-kesalahan yang menunjukan ketidak- sesuaian denga persyaratan pelanggan. Setiap

fungsi aplikasi web dinilai dalam aspek-aspek yang terkait dengan kebenaran, ketidakstabilan dan kesesuaian umum dengan standar pelaksaan yang sesuai (misal : standar bahasa Java atau AJAX)

Struktur, dinilai untuk memastikan bahwa aplikasi web tersebut benar-benar menyediakan isi dan fungsi aplikasi web, bahwa struktur dapat diperluas dan dapat didukung saat isi atau fungsionalitas yang baru ditambahkan.

Kegunaan, diuji untuk memastikan bahwa setiap kategori pengguna didukung oleh antarmuka dan dapat belajar menerapkan semua sintaks dan semantik navigasi yang diperlukan

Kemampuan untuk dapat dinavigasi, diuji untuk memastikan bahwa semua sintaks dan semantik navigasi dilakukan untuk menemukan kesalahan navigasi apapun (misal, tautan mati/dead link, tautan yang tidak benar, tautan yang salah)

Kinerja, diuji di bawah berbagai kondisi operasi, konfigurasi dan pemuatan/loading untuk memastikan bahwa sistem responsif terhadap interaksi pengguna dan dapat menangani beban ekstrem tanpa menurunkan kemampuan operasional yang tidak dapat diterima.

Kompatibilitas, diuji dengan menjalankan aplikasi web dalam berbagai konfigurasi host yang berbeda abik apda sisi klien amupun server. Tujuannya untuk menemukan kesalahan yang khusus pada konfigurasi host yang unik

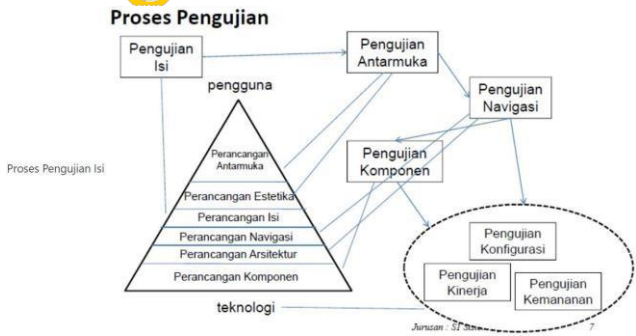
Interoperabilitas, diuji untuk memastikan bahwa apkikasi web berantarmuka dengan benar dengan aplikasi lain dan/atau basis data

Keamanan, diuji dengan menilai kerentanan potensial dan berusaha menyinkap masing- masing kerentanan. Setiap usaha penetrasi yang sukses dianggap sebagai suatu kegagalan keamanan

Strategi Pengujian Aplikasi Web

Strategi Pengujian Aplikasi Web

1. Model konten untuk aplikasi web ditinjau untuk menemukan kesalahan
2. Model antarmuka ditinjau untuk memastikan bahwa semua use case dapat diakomodasi
3. Model perancangan untuk aplikasi web ditinjau untuk mengungkap kesalahan navigasi
4. Antarmuka pengguna diuji untuk mengungkap kesalahan dalam presentasi dan / atau mekanik navigasi
5. Komponen fungsional diuji untuk setiap unit
6. Navigasi seluruh arsitektur diuji Aplikasi web diimplementasikan dalam berbagai konfigurasi lingkungan yang berbeda dan diuji kompatibilitasnya pada masing-masing konfigurasi
7. Pengujian keamanan dilakukan dalam upaya untuk menyinkapkan kelemahan-kelemahan dalam apliaksi web atau kelemahan dalam lingkungannya.
8. Pengujian kinerja dilakukan
9. Aplikasi web diuji oleh populasi pengguna akhir yang dikontrol dan dipantau; hasil interaksi mereka denga sistem kemudian dievaluasi untuk menemukan kesalahan isi dan navigasi, kegunaan-kegunaan penting, kesesuaian/compatibility, kemamanan, keandalan dan kinerja aplikasi web



Gambar 6. 1 Proses Pengujian

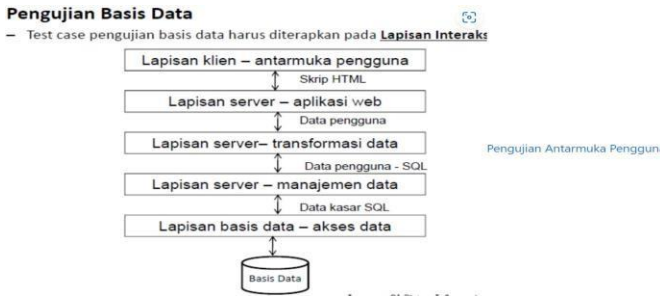
Proses Pengujian

Tujuan :

1. Mengungkap kesalahan sintaksis (misal : kesalahan ketik, kesalahan tata bahasa) dalam dokumen berbasi s teks, representasi grafis dan media lainnya
2. Mengungkap kesalahan-kesalahan semantik (kesalahan dalam ketepatan atau kelengkapan informasi) di sembarang isi objek yang disajikan saat navigasi terjadi
3. Mencari kesalahan-kesalahan dalam pengaturan atau struktur isi yang disajikan kepada pengguna akhir.
4. Peninjau / Tester harus menjawab pertanyaan-pertanyaan berikut :
5. Apakah informasi faktual akurat ?
6. Apakah informasi ringkas dan langsung menuju sasaran ?
7. Apakah informasi yang dimuat dalam objekisi dapat ditemukan dengan mudah ?
8. Apakah referensi yang alyak telah disediakan untuk semua informasi yang diperoleh dari sumber lainnya ?
9. Apakah informasi yang disajikan konsisten secara internal dan konsisten dengan informasi yang disajikan pada objek lainnya ?
10. Apakah isi bersifat menyerang, menyesatkan atau

membuka pintu munculnya kasus hukum ?

11. Apakah isi melanggar hak cipta atau merk dagang yang ada ?
12. Apakah isi memuat tautan-tautan internal yang melengkapi isi yang sudah ada ? Apakah tautan tersebut sudah benar ?
13. Apakah gaya estetika isi bertentangan denga gaya estetika antarmuka ?



Gambar 6. 2 Pengujian Basis data

Pengujian antarmuka menguji mekamisme-mekanisme interaksi dan memvalidasi aspke-aspek estetika dari pengguna antarmuka.
Tujuan pengujian antarmuka pengguna :

1. Mengungkap kesalahan yang terkait dengan mekanisme antaramuka tertentu (misal kesalahan dalam mengesekusi denga benar sebah link menu atau cara memasukan data pada sebuah formulir)
2. Menemukan kesalahan-kesalahan dalam cara antarmuka menerapkan semantik navigasi, fungsi aplikasi web atau tampilan isi. Strategi pengujian antarmuka :
3. Fitur-fitur antarmuka diuji untuk memastikan bahwa aturan perancangan, estetika dan isi vitual terkait
4. yang tersedia pengguna tidak mengandung kesalahan seperti fitur : jenis huruf penggunaan Tekni

5. Bagi kesalahan, huruf, warna, bingkai, gambar, garis tepi, tabel dan fitur antarmuka lainya yang terkait dengan hasil eksekusi Aplikasi web
6. Masing-masing mekanisme antarmuka diuji dnegan cara yang sama dengan pengujian unit.

Pengujian Antarmuka Pengguna

Pengujian antarmuka menguji mekamisme-mekanisme interaksi dan memvalidasi aspke-aspek estetika dari pengguna antarmuka.

Tujuan pengujian antarmuka pengguna :

- Mengungkap kesalahan yang terkait dengan mekanisme antaramuka tertentu (misal kesalahan dalam mengesekusi denga benar sebah link menu atau cara memasukan data pada sebuah formulir)
- Menemukan kesalahan-kesalahan dalam cara antarmuka menerapkan semantik navigasi, fungsi aplikasi web atau tampilan isi.

Strategi pengujian antarmuka :

- Fitur-fitur antarmuka diuji untuk memastikan bahwa aturan perancangan, estetika dan isi vitual terkait yang tersedia bagi pengguna tidak mengandung kesalahan, seperti fitur : jenis huruf, penggunaan warna, bingkai, gambar, garis tepi, tabel dan fitur antarmuka lainya yang terkait dengan hasil eksekusi aplikasi web
- Masing-masing mekanisme antarmuka diuji dnegan cara yang sama dengan pengujian unit. Misal
- pengujian yangdirancang untuk mencalankan semua form, penulisan skrip sisiklien, HTML dinamis, skrip, isi streaming dan

aplikasi mekanisme spesifik antarmuka (misal keranjang belanja pada aplikasi e- commerce)

- Setiap mekanisme antarmuka diuji dalam konteks penggunaan use case untuk kategori pengguna tertentu.
- Antarmuka diuji dalam lingkungan berbagai lingkungan perambah/browser untuk memastikan bahwa antarmuka tersebut sesuai.

Mekanisme Pengujian antarmuka :

- Tautan, setiap tautan diuji untuk memastikan bahwa objek isi atau fungsi yang tepat tercapai.Formulir, memastikan bahwalabel mengidentifikasi dengan benar bidang dalam formulir dan bidan wajib diidentifikasi secara visual bagi pengguna,server menerima semua informasi yang ada dalam form dan tidak ada data yang hilang saat terjadi transmisi antara klien dan server,digunakan default yang tepat saat pengguna tidak memilih dari menu pull down atau serangkaian tombol,fungsi-fungsi perambah (misal : tombol back) tidak merusak data yang diisikan ke dalam form,skrip yang melakukan pemeriksaan kesalahan pada data yang dimasukan, bekerja dengan baik dan memberikanpesan kesalahanyang signifikan client-side scripting, pengujian untuk menemukan kesalahan kesalahan dalam pengolahan saat

Pengujian Konfigurasi

Pengujian Konfigurasi

Konfigurasi variabilitas dan ketidakstabilan adalah faktor penting dalam pengujian aplikasi web.Pengujian konfigurasi di sisi server :

- Apakah aplikasi web sepenuhnya kompatibel denga server OS ?
- Apakah berkas-berkas sistem, direktori dan data sistem

yang terkait dibuat dengan bensar saat aplikasi web tersebut operasional ?

- Apakah ukuran kemananan sistem (firewall/eknkripsi) mengijinkan apaliaksi web berjalan melalyani pengguna tanpa gangguan penurunan kinerja ?
- Apakah aplikasi web telah diuji dengan konfigurasi jika ada server yang didistribusikan ?
- Apakah aplikasi web terintegrasi secara tepat denga PL basis data ? Apkah aplikasi web sensitif terhadap versi PL basis data yang berbeda-beda ?
- Apakah skrip aplikasi web sisi server mengeksekusi dengan benar ?
- Apakah kesalahan administrator sitem telah diuji efeknya pada operasi aplikasi web

Masalah-masalah di sisi klien :

- Perangkat keras
- Sistem Operasi
- Browser
- Komponen antarmuka pengguna (Active-X, Java applet)
- Plug in (Quick Time, Real Player)
- Konektivitas (kabel modem)

Pengujian Keamanan

Pengujian keamanan dirancang untuk menyelidiki kerentanan lingkungan sisi klien, komunikasi jariangan yang terjadi saat data dilewatkan dari klien ke server dan kembali lagi dan lingkungan sisi server.

Pada sisi klien kerentanan dilacak pada bug yang telah ada sebelumnya pada browser, email program ,PL komunikasi, akses tidak sah ke cookie yang ditempatkan pada browser.

Pada sisi server kerentanan meliputi serangan DOS (Denial of service) dan skrip jahat yang diteruskan ke sisi klien atau diguankan untuk mematahkan operasi server

Perlindungan keamanan :

1. Firewall – mekanisme penyaringan yang merupakan kombinasi dari perangkat keras dan perangkat lunak yang memeriksa setiap paket informasi yang datang untuk memastikan bahwa informasi tersebut berasal dari sumber yang sah, memblokir data yang dicurigai.
2. Otentifikasi – mekanisme verifikasi identtias yang menvalidasi semua klien dan server, yang memungkinkan komunikasi terjadi hanya bila kedu abelah pihak telah diverifikasi.
3. Enkripsi – mekanisme penyandian yang melindungi data sensitif dengancara memodifikasi data dengan cara yang tidak memungkinkan data dibacara oleh orang-orang yang berniat jahat
4. Otorisasi – mekanisme penyairngan yang memungkinkan akses ke klien atau lingkungan server hanya untuk orang-orang dengan kode otoritas yang tepat.

Pengujian Kinerja

Pengujian Kinerja

Tujuan : mensimulasikan situasi-situasi pemuatan (loading) yang sesungguhnya, yaitu pada saat jumlah pengguna aplikasi bertambah atau umlah transaksi online meningkat atau jumlah data meningkat

Pertanyaan yang diajukan :

- Apakah waktu tanggap server turun ke titik dimana itu nyata dan tidak dapat diterima ?
- Apakah komponen sistem bertanggungjawab atas penurunan kinerja ?

- Apakah waktu tanggap rata-rata bagi para pengguna berada di bawah berbagai kondisi pemuatan?
- Apakah degradasi kinerja berdampak pada keamanan sitem ?
- Apakah keandalan atau ketapatanaplikasi web terpengaruh saat loading ke sistem bertambah ?

Pengujian Pemuatan
Tujuan :
Menentukan bagaiman aplikasi web dan lingkungan sisi server akan Manggapi berbagai kondisi pemuatan, variabel yang digunakan adalah
N, jumlah pengguna yang melakukan loading secara bersamaan
T, jumlah transaksi online per unit waktu

PENGUJIAN APLIKASI BERBASIS MOBILE

Pengujian aplikasi berbasis mobile

Testing aplikasi mobile adalah proses dimana aplikasi yang dikembangkan untuk perangkat mobile diuji untuk kegunaan dan konsistensinya. Ada dua macam pengujian yang harus dilakukan untuk aplikasi pada perangkat mobile yaitu pengujian hardware dan software, dan berikut adalah pembahasanya.

Testing hardware

Hardware termasuk prosesor, memori internal, ukuran layar, resolusinya, besarnya RAM, kemampuan kamera, bluethooth, WIFI dan lain-lain akan menjadi komponen yang yang harus diuji dalam proses pengujian ini.

Testing Software

Dalam tahap ini kemudian aplikasi mobile yang telah dikembangkan akan diuji secara mendetail dan teliti. Fungsi dan konsistenti sebuah aplikasi mbile akan diuji. Seperti kita ketahui bahwa aplikasi mobile terbagi menjadi tiga macam yakni: aplikasi mobile native, aplikasi mobile hybrid dan aplikasi mobile web. Ketiga nya memiliki perbedaan dasar yang tentunya akan mempengaruhi proses pengujian.

Pengujian aplikasi mobile jauh lebih kompleks dibanding pengujian aplikasi atau web untuk dekstop karena:

Perangkat mobile memiliki banyak macam ukuran layar

konfigurasi hardware seperti keypad, virtual keypad (touch screen), trackball dan lain-lain.

Bermacam-macam sistem operasi yang di pakai, seperti android, windows, blackberry, dan iOS Bermacam-macam versi dari sistem operasi

Bermacam-macam jenis jaringan mobile, seperti CDMA atau GSM, berkemampuan EDGE, 3G, atau 4G.

Untuk mengatasi semua masalah teknis di atas, ada beberapa macam testing yang selayaknya dilakukan pada aplikasi mobile.

Usability testing, untuk memastikan bahwa aplikasi mobile mudah digunakan dan memberikan user experience yang baik untuk penggunanya.

Compatibility testing, pengujian aplikasi dengan perangkat mobile yang berbeda, melalui browser, dengan ukuran layar yang berbeda serta versi OS sesuai dengan kebutuhan.

Interface testing, pengujian pilihan menu, tombol, bookmark, history, pengaturan, dan anvigasi dari aplikasi

Service testing, pengujian aplikasi dalam keadaan online maupun offline

Low leve resource testing, pengujian memori, auto-delete file-file sementara, masalah pertumbuhan database.

Performance testing, pengujian kinerja aplikasi dengan mengubah koneksi dari 2G atau 3G ke wifi. Bagaimana kemampuan berbagi dokumennya dan bagaimana kapasitas baaterai yang dibutuhkan.

Operational testing, backup dan rencana recovery jika baterai melemah atau saat kehilangan data karena proses upgrade dari toko aplikasi.

Installation testing, validasi aplikasi dengan menginstall atau menguninstall pada perangkat mobile Securuty testing, pengujian aplikasi untuk memvalidasi apakah data terlindungi system informasi.

Untuk memastikan bahwa semua standart kualitas dan kinerja terpenuhi maka adanya strategi untuk testing aplikasi mobile sangat dibutuhkan.

Pemilihan perangkat, menganalisis pasar dan memilih perangkat yang banyak digunakan.
Keputusan ini sebagian besar bergantung pada pengguna, atau pada pengembang yang mempertimbangkan faktor popularitas sebuah perangkat tertentu.)
Emulator, penggunaan emolator sangat membantu dalam tahap pengembangan awal. Emulator memungkinkan untuk melalukukan pengecekan secara cepat dan efisien. Emulator sendiri merupakan sistem yang menjalankan software yang seolah-olah membawa kita pada sistem operasi lain.

Macam-macam emulator mobile:
Device emulator – disediakan oleh produsen perangkat yang terkait
Browser emulator – mensimulasikan kondisi mobile browser
Sistem operasi emulator – Apple menyediakan emulator untuk iPhone, m
icrosoft untuk ponsel windows dan Android untuk windows.
Mempertimbangkan cloud computing based testing, menciptakan kondisi mobile berbasis web pada emulayor untuk mengakses aplikasi mobile.
Automation vs. Manual testing
Jika aplikasi mengusung fungsi baru, maka perlu dilakukan pengujian manual.

Jika aplikasi membutuhkan pengujian sekali atau dua kali, akan lebih baik jika dilakukan secara manual
Mengotomasi script untuk kasus uji regresi. Jika tes regesi perlu

dilakukan berulang, pengujian otomatis sangat cocok untuk ini. Dua jenis tools otomatisasi yang tersedia untuk menguji aplikasi mobile

Object based mobile testing tools – otomatisasi oleh unsur-unsur pemetaan pada layar perangkat yang di targetkan.

Image based mobile testing tools – membuat skrip otomatisasi berdasarkan koordinat elemen layar.

Network configuration, bagian sangat diperlukan pada proses pengujian aplikasi mobile yakni untuk memvalidasi aplikasi pada kemampuan jaringan yang berbeda seperti 2G, 3G, 4G atau wifi.

Kesimpulannya adalah, proses pengujian aplikasi mobile membutuhkan strategi tes yang tepat. Memilih simulator mobile, memilih perangkat dan tools testing yang tepat dapat membantu Anda memastikan memiliki 100% cakupan pengujian aplikasi mobile yang meliputi pengujian keamanan, kegunaan, kinerja, fungsi dan kompatibilitas. Pastikan aplikasi mobile Anda memiliki hasil pengujian yang baik sebelum sampai pada penggunanya.

Tahukah kamu apa perbedaan antara aplikasi native, hybrid, dan web?

Ya, ketiganya merupakan aplikasi yang bisa kamu akses di smartphone. Namun, ketiganya memiliki perbedaan yang cukup signifikan dari berbagai sisi.

Aplikasi Native

Sebelum mengetahui perbedaan aplikasi native, hybrid, dan web, ada baiknya kita memahami definisi dari masing-masing aplikasi tersebut.

Dilansir dari Codepolitan, aplikasi native adalah aplikasi yang dibangun dengan bahasa pemrograman yang spesifik dan hanya dapat digunakan di platform tertentu. Aplikasi native bisa juga disebut aplikasi asli.Untuk membuat aplikasi di dua sistem operasi yang berbeda, kamu membutuhkan bahasa pemrograman yang

berbeda pula. Sebagai contoh, kamu bisa menggunakan bahasa pemrograman Objective-C atau Swift untuk iOs. Sementara itu, kamu bisa menggunakan bahasa pemrograman Java untuk platform Android.

Adapun kedua sistem operasi tersebut memiliki integrated development environment (IDE). IDE yang digunakan Android adalah Android Studio. Sementara itu, iOS menggunakan IDE iOS Xcode. Untuk menggunakan aplikasi native, pengguna bisa men-download-nya langsung dari Google Play Store, App Store, dan sejenisnya.

Salah satu kelebihan dari aplikasi native adalah UI/UX-nya yang sangat baik. Sayangnya, aplikasi ini hanya dapat digunakan di satu platform dan biaya pengembangannya pun relatif tinggi.

Perbedaan native, hibryd, dan mobile web application

Aplikasi Hybrid

Salah satu perbedaan mencolok antara aplikasi native, hybrid, dan web adalah dari segi platform yang digunakan.

Jika aplikasi native hanya bisa digunakan di salah satu platform, aplikasi hybrid justru dapat digunakan di berbagai platform.

Secara definisi, aplikasi hybrid adalah aplikasi yang dibuat menggunakan bahasa pemrograman web dengan bantuan software development kit (SDK) native dari berbagai platform sistem operasi.

Aplikasi hybrid menggabungkan elemen aplikasi native dan web, seperti ditulis Gist. Selain itu, jenis aplikasi ini juga menggabungkan berbagai fitur sistem operasi.

Jadi jika ingin aplikasimu dapat digunakan di Android, iOs dan sistem operasi lainnya, kamu bisa memilih aplikasi hybrid.

Layaknya aplikasi native, aplikasi hybrid juga dapat di-download di berbagai app store. Salah satu kelebihan aplikasi hybrid adalah biasanya lebih mudah dan lebih cepat untuk dikembangkan. Aplikasi ini juga membutuhkan lebih sedikit maintenance atau perawatan. Namun, umumnya performa aplikasi hybrid belum bisa mengungguli aplikasi native.

Aplikasi Web

Pernahkah kamu mengakses suatu situs yang berawalan huruf "m"? Misalnya, "m.facebook.com" atau "m.tokopedia.com".

Nah, itulah yang disebut aplikasi web. Dilansir dari MobiLoud, aplikasi web adalah aplikasi yang diakses melalui web browser dengan menggunakan koneksi internet.Dengan demikian, kamu bisa melihat satu lagi perbedaan aplikasi native, hybrid, dan web. Jika aplikasi native dan hybrid harus di-download di app store, tidak dengan aplikasi web. Aplikasi ini dapat digunakan hanya dengan mengakses web browser.Kebanyakan aplikasi web dikembangkan menggunakan JavaScript, CSS dan HTML5. Aplikasi yang satu ini juga tidak memerlukan SDK layaknya aplikasi Android dan iOS. Lalu, apa keunggulan aplikasi web? Aplikasi ini hanya menggunakan teknologi web. Maka, pengembangannya pun relatif lebih mudah daripada jenis aplikasi lainnya. Sayangnya, kemampuan aplikasi web terbatas dan tidak bisa di-publish ke berbagai app store. Baca Juga: Mengenal iOS Developer, Sosok di Balik Aplikasi Perangkat Keluaran Apple. Itulah penjelasan Glints tentang perbedaan aplikasi native, hybrid, dan web. Sekarang kamu sudah lebih paham, bukan? Intinya, tiga jenis aplikasi tersebut memiliki perbedaan dasar dari segi platform dan pengembangan. Kamu bisa membuat aplikasi yang sesuai dengan kebutuhanmu. Setelah memahami perbedaan aplikasi native, hybrid, dan web, apakah kamu tertarik untuk membangun karier sebagai pengembang aplikasi?.

Kategori pengujian pada aplikasi berbasis web mobile

Tahapan Terpenting Dalam Pengujian Aplikasi Mobile

QUADRA SOLUTION – Testing Tool Ternyata ada yang menarik bila Anda menilik lebih jauh apa dan bagaimana sebuah aplikasi mobile itu dibuat. Selain membutuhkan human resource, seperti tenaga, pikiran, dan tentunya support aplikasinya. Pembuatan perangkat lunak berbasis mobile ternyata juga memiliki nilai insentif yang berbeda-beda untuk setiap platform.

Misalnya, seorang developer Android dan iOS. Dilihat dari background developer yang dibutuhkan sudah cukup berbeda, selain itu biaya yang dibutuhkan dalam membuat aplikasi dalam kedua platform tersebut juga berbeda. Hal pertama yang membedakan pembiayaan development aplikasi mobile adalah perangkat yang dibutuhkan dan support aplikasi untuk keperluan development tersebut.

Pada dasarnya tujuan utama dibuatnya aplikasi berbasis mobile adalah untuk memberikan pengalaman baru kepada penggunanya serta dapat memanfaatkannya secara lebih ringkas, cepat, dan tentu mudah dalam pemakainya. Dalam membuat aplikasi berbasis mobile tentu saja Anda tidak akan terlepas dengan adanya “bug” atau kesalahan “error” pada proses-proses tertentu Maka dari itu, untuk menghindari banyaknya “bug” atau “error” maka sangat diperlukan dan dilakukan pengujian sebelum aplikasi mobile tersebut diberikan ke customer atau selama aplikasi mobile masih terus dikembangkan. Pada dasarnya pengujian ini sangatlah penting dan sangat mengacu pada kualitas aplikasi mobile tersebut.

Berikut ini adalah 4 (empat) tahapan terpenting dalam pengujian aplikasi mobile:

Pengujian Unit/Unit Testing. Unit testing adalah metode verifikasi pada perangkat lunak di mana programmer akan menguji suatu unit program apakah layak atau tidak untuk digunakan. Unit testing ini fokusnya pada verifikasi pada unit yang terkecil pada desain perangkat lunak (komponen atau modul perangkat lunak). Karena dalam sebuah perangkat lunak banyak memiliki unit-unit kecil maka untuk mengujinya biasanya dibuat program kecil atau main program) untuk menguji unit-unit perangkat lunak tersebut.

Pengujian Integrasi. Pengujian integrasi lebih kepada pengujian penggabungan dari dua atau lebih unit pada perangkat lunak. Pengujian integrasi ada baiknya dilakukan secara bertahap untuk menghindari kesulitan apabila terjadi kesalahan erroratau “bug”

Baca juga Bagaimana Membuat Testing Tool Bekerja Lebih Cepat dan Efisien?

Pengujian Sistem. Unit-unit proses yang telah diintegrasikan diuji dengan antarmuka yang sudah dibuat sehingga pengujian ini dimaksud untuk menguji system perangkat lunak menggunakan Testing Tool dan perlu diingat bahwa dalam melakukan pengujian system harus dilakukan secara bertahap sejak awal pengembangan, apabila dalam pengujian hanya diakhir maka dapat dipastikan kualitas dari sistem yang bagus

Pengujian Penerimaan. Pengujian penerimaan perangkat lunak dilakukan oleh pengguna yang telah bekerja sama dengan pembuat program guna untuk mengetahui secara langsung bagaimana perangkat lunak tersebut dapat berkerja secara normal sebelum disebarluaskan untuk customer. Pengujian penerimaan ini memiliki tujuan untuk mengetahui kepuasan dari pengguna atau user.

Intinya, pengujian pada perangkat lunak merupakan elemen kritis

dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean.

Pengujian merupakan suatu kegiatan dimana suatu sistem atau komponen yang dijalankan dalam kondisi tertentu, yang mana hasilnya diamati dan direkam untuk kemudian dilakukan evaluasi (IEEE Standard Glossary).

Jenis pengujian pada palikasi berbasisweb

Apa itu Pengujian Web?

Pengujian Web secara sederhana memeriksa aplikasi web Anda untuk kemungkinan bug sebelum dibuat langsung atau sebelum kode dipindahkan ke lingkungan produksi.

Selama tahap ini masalah seperti keamanan aplikasi web, fungsi situs, aksesnya kepada pengguna yang cacat serta pengguna reguler dan kemampuannya untuk menangani lalu lintas diperiksa.

Cara menguji Aplikasi Web

Dalam Rekayasa Perangkat Lunak, jenis / teknik pengujian berikut dapat dilakukan tergantung pada persyaratan pengujian web Anda.

1. Pengujian Fungsionalitas:

Pengujian ini digunakan untuk memeriksa apakah produk Anda sesuai dengan spesifikasi yang Anda maksudkan serta persyaratan fungsional yang Anda buat dalam dokumentasi pengembangan Anda. Kegiatan Pengujian Berbasis Web meliputiUji semua tautan di laman web Anda berfungsi dengan benar dan pastikan tidak ada tautan yang rusak. Tautan yang akan diperiksa akan mencakup – Tautan keluar Tautan internal Anchor Links Tautan MailToTest Form berfungsi seperti yang diharapkan. Ini akan mencakup- Pemeriksaan skrip pada formulir berfungsi seperti yang

diharapkan. Misalnya – jika pengguna tidak mengisi bidang wajib dalam bentukpesan kesalahan ditampilkan.Periksa nilai default sedang diisi. Setelah dikirimkan, data dalam formulir dikirimkan ke database langsung atau ditautkan ke alamat email yang berfungsi.

Formulir diformat secara optimal untuk keterbacaan yang lebih baik Test Cookies berfungsi seperti yang diharapkan. Cookie adalah file kecil yang digunakan oleh situs web untuk mengingat sesi pengguna aktif sehingga Anda tidak perlu login setiap kali Anda mengunjungi situs web. Pengujian Cookie akan mencakup Cookie pengujian (sesi) dihapus baik ketika cache dihapus atau ketika mereka mencapai

2. Usability testing:

Pengujian Kegunaan kini telah menjadi bagian penting dari setiap proyek berbasis web. Ini dapat dilakukan oleh penguji seperti Anda atau kelompok fokus kecil yang mirip dengan audiens target aplikasi web.

Uji Navigasi situs:

Menu, tombol atau Tautan ke berbagai halaman di situs Anda harus mudah dilihat dan konsisten di semua halaman web

Uji Konten:

Konten harus dapat dibaca tanpa kesalahan ejaan atau tata bahasa. Gambar jika ada harus berisi teks “alt”

Alat yang dapat digunakan: Chalkmark, Clicktale, Clixpy and Feedback Army

3. Interface Testing:

Tiga area yang akan diuji di sini adalah – Aplikasi, Web, dan Server Database

Aplikasi: Permintaan tes dikirim dengan benar ke Database dan output di sisi klien ditampilkan dengan benar. Kesalahan jika ada harus ditangkap oleh aplikasi dan harus hanya ditampilkan kepada administrator dan bukan pengguna akhir.

4. Database Testing:

Basis data adalah salah satu komponen penting dari aplikasi web Anda dan tekanan harus diletakkan untuk mengujinya secara menyeluruh. Kegiatan pengujian akan mencakup- Uji apakah ada kesalahan yang ditampilkan saat menjalankan query Integritas Data dipertahankan saat membuat, memperbarui ataumenghapus data dalam basis data. Periksa waktu respons pertanyaan dan sesuaikan jika perlu. Data uji yang diambil dari basis data Anda ditampilkan secara akurat di aplikasi web AndaAlat yang dapat digunakan: QTP, Selenium.

5. Compatibility testing.

Tes kompatibilitas memastikan bahwa aplikasi web Anda ditampilkan dengan benar di berbagai perangkat. Ini akan mencakup-Uji Kompatibilitas Browser: Situs web yang sama di browser yang berbeda akan ditampilkan secara berbeda. Anda perlu menguji apakah aplikasi web Anda ditampilkan dengan benar di seluruh browser, JavaScript, AJAX dan otentikasi berfungsi dengan baik. Anda juga dapat memeriksa Kompatibilitas Browser SelulerRender elemen web seperti tombol, bidang teks dll berubah seiring perubahan Sistem Operasi. Pastikan situs web Anda berfungsi dengan baik untuk berbagai kombinasi sistem Operasi seperti Windows, Linux, Mac dan Browser seperti Firefox, Internet Explorer, Safari dll.Alat yang dapat digunakan: NetMechanic.

6. Performance Testing:

   Ini akan memastikan situs Anda berfungsi di bawah semua beban. Kegiatan Pengujian Perangkat Lunak akan termasuk tetapi tidak terbatas pada Waktu respons aplikasi situs web pada kecepatan koneksi yang berbeda Muat uji aplikasi web

Anda untuk mengetahui perilakunya di bawah beban normal dan puncak Uji stres situs web Anda untuk menentukan titik putus ketika didorong ke luar beban normal pada waktu puncak. Uji apakah terjadi kerusakan karena beban puncak, bagaimana situs pulih dari peristiwa semacam itu. Pastikan teknik pengoptimalan seperti kompresi gzip, browser, dan cache sisi server diaktifkan untuk mengurangi waktu muat. Alat yang dapat digunakan: LoadRunner, Jmeter.

7. Security Testing
   Pengujian Keamanan sangat penting untuk situs web e-commerce yang menyimpan informasi pelanggan yang sensitif seperti kartu kredit. Kegiatan Pengujian akan mencakup-tes akses tidak sah ke halaman aman tidak boleh diizinkan file yang dibatasi tidak boleh diunduh tanpa akses yang sesuai sesi pemeriksaan secara otomatis terbunuh setelah pengguna tidak aktif dalam waktu lama Pada penggunaan sertifikat SSL, situs web harus mengarahkan kembali ke halaman SSL terenkripsi. Alat yang dapat digunakan: Babel Enterprise, BFBTester dan CROSS
8. Crowd Testing:
   Anda akan memilih sejumlah besar orang (kerumunan) untuk melakukan tes yang jika tidak akan dieksekusi sekelompok orang di perusahaan. Pengujian crowdsourced adalah konsep yang menarik dan akan datang dan membantu mengungkap banyak kekurangan tanpa disadari. Ini menyimpulkan tutorial. Ini mencakup hampir semua jenis pengujian yang berlaku untuk aplikasi web Anda.

STANDAR PROGRAM YANG BAIK

Standar pemrograman dibutuhkan untuk menciptakan suatu program dengan portabilitas yang tinggi sehingga memudahkan

dalam merancang dan merawat program serta meningkatkan efektivitas penggunaan peralatan komputer. Beberapa standar dasar penilaian untuk sebuah program dikatakan baik antara lain:
Teknik pemecahan masalah
Penyusunan program
Perawatan program
Standar prosedurs

STANDAR TEKNIK PEMECAHAN MASALAH
Setelah masalah dipahami dengan baik, seorang pemrograman membutuhkan suatu teknik untuk memecahkan masalah tersebut.

Ada dua pendekatan yang umum digunakan, yakni:
Teknik Top-Down merupakan teknik pemecahan masalah di mana suatu masalah yang kompleks dibagi-bagi menjadi beberapa struktur hingga unit yang paling kecil, setelah itu kemudian disusun langkahlangkah untuk menyelesaikan masalah secara rinci. Teknik semacam ini digunakan pada metode pemrograman terstruktur
Teknik Bottom-Up merupakan teknik pemecahan masalah yang berkebalikan dengan teknik Top- Down di mana penyelesaian masalah dimulai dari hal-hal yang bersifat khusus, kemudian naik ke bagian yangbersifat umum. Teknik semacam ini digunakan pada metode pemrograman berorientasi objek
Setelah memilih teknik pemecahan masalah, pemrogram mulai menyusun langkah-langkah untuk memecahkan masalah, yang disebut dengan algoritma. Algoritma yang baik memiliki ciri-ciri sebagaiberikut:

a. Tepat, benar, sederhana, standar, dan efektif
b. Logis, terstruktur, dan sistematis
c. Semua operasi terdefinisi
d. Semua proses harus berakhir setelah sejumlah langkah dilakukan
e. Menggunakan bahasa standar sehingga tidak ambigu

STANDAR PENYUSUNAN PROGRAM

Beberapa faktor yang menjadi standar dalam penyusunan program antara lain:

Kebenaran logika dan penulisan

Program yang disusun harus memiliki kebenaran logika dalam pemecahan masalah maupun penulisan kode program. Program harus tepat dan teliti dalam perhitungan sehingga hasilnya dapat dipercaya

Waktu minimum untuk penulisan program

Penulisan program harus memiliki waktu minimum, artinya waktu minimal yang harus tersedia untuk menuliskan kode program dari awal hingga siap untuk dieksekusi

Kecepatan maksimum eksekusi program

Agar program memiliki kecepatan eksekusi maksimum, perlu diperhatikan beberapa hal antara lain bahasa pemrograman yang digunakan, algoritma yang disusun, teknik pemrograman yang dipakai, dan perangkat keras yang digunakan. Kecepatan maksimum juga dapat ditingkatkan dengan memperbaiki struktur program misalkan:

Menghindari proses pengujian yang berulang-ulang secara percuma

Meletakkan syarat pengujian yang akan menolak data dengan jumlah terbanyak sebagai syarat pengujian pertama, syarat pengujian dengan jumlah terbanyak kedua sebagai syarat pengujiankedua, dan seterusnya

Memperbaiki susunan baris program guna meningkatkan kecepatan eksekusi

Ekspresi penggunaan memori

Semakin sedikit penggunaan memori, semakin cepat program dieksekusi. Untuk meminimumkan penggunaan memori, maka perlu diperhatikan:

1. Menggunakan tipe data yang cocok untuk kebutuhan pemrograman
2. Menghindari penggunaan variabel berindeks secara berulang kali
3. Kemudahan merawat dan mengembangkan program
4. Program yang memiliki struktur yang baik, struktur data jelas, dan dokumentasi yang lengkap dan mudah dipahami, akan mudah untuk dirawat dan dikembangkan
5. User friendly
6. Program yang baik harus memiliki layanan untuk mempermudah pemakai untuk menggunakannya, misalkan layanan online help
7. Portabilitas
8. Program yang baik harus dapat dijalankan pada kondisi platform yang berbeda-beda, baik itu sistem operasi maupun perangkat keras.

Modular 3

Pada pendekatan pemrograman, masalah dibagi-bagi menjadi unit terkecil, yang disebut modul untuk menyederhanakan pengimplementasian langkah-langkah pemecahan masalah dalam bentuk program

A.2.1.3 STANDAR PERAWATAN PROGRAM

Beberapa standar yang harus dipenuhi agar memudahkan pemrogram dalam merawat dan mengembangkan program antara lain:

Dokumentasi

Dokumentasi merupakan catatan dari setiap langkah pekerjaan membuat program dari awal hingga akhir. Dokumentasi ini penting untuk memudahkan menelusuri adanya kesalahan maupun untuk pengembangannya. Dokumentasi yang baik akan memberikan informasi yang menanda

1. Penulisan Instruksi
2. Untuk memudahkan perawatan program, sebaiknya penulisan program dilakukan sebagai berikut:
   - Menuliskan satu instruksi pada satu baris program
   - Memisahkan modul-modul dengan memberikan spasi beberapa baris untukmempermudah pembacaan
   - Membedakan bentuk huruf dalam penulisan program
   - Memberikan tabulasi yang berbeda untuk penulisan instruksiinstruksi yang berada pada loop atau struktur kondisional
   - Menghindari penggunaan konstanta dalam penulisan rumus, jika konstanta tersebut mungkin berubah
   - Melakukan pembatasan jumlah baris instruksi per moduli sehingga orang lain dapat mengerti dan memahami alur logika program.

A.2.1.4 STANDAR PROSEDUR

Penggunaan prosedur standar akan memudahkan bagi

pengembang program untuk mengembangkan program tersebut

MODULARITAS

Modularitas merupakan sebuah konsep untuk memecah program menjadi modul-modul kecil di mana masing-masing modul berinteraksi melalui antarmuka modul. Dengan adanya modularitas, kesalahan di satu bagian program dapat dikoreksi tanpa perlu mempertimbangkan bagian-bagian lainnya, program menjadi lebih sederhana sehingga lebih mudah dipahami.

KRITERIA MODULARITAS

Terdapat lima kriteria modularitas, yakni:

1. Decomposibility Kemampuan untuk mendekomposisi masalah menjadi submasalah yang lebih sederhana dan dihubungkan dengan struktur yang sederhana
2. Composability Kemampuan membangun modul-modul program yang kemudian dapat diintegrasikan menjadi program pada lingkungan yang mungkin berbeda dengan saat modul tersebut dibangun.
3. Understandability Kemampuan menghasilkan program di mana programmer dapat memahami masing-masing modul tanpa perlu mengetahui detailnya
4. Continuity Kemampuan meredam propagasi perubahan, yaitu suatu kondisi di mana perubahan kecil pada satu modul memicu perubahan hanya pada satu modul atau sedikit modul yang terkait
5. Protection Kemampuan meredam kondisi abnormal hanya pada satu modul.

ATURAN MODULARITAS

Terdapat pula lima aturan modularitas, antara lain:

1. Direct mapping
   Struktur model yang ada pada masing-masing tahap pengembangan perangkat lunak semestinya kontinyu, dalam artian modul yang terdapat pada analisis masih merupakan modul pada tahap perancangan dan tetap menjadi modul pada saat pemrograman.
2. Few interfaces
   Setiap modul seharusnya berinteraksi dengan sesedikit mungkin dengan modul lain sebab jika terjadi banyak interaksi antar modul akan meningkatkan propagasi perubahan
3. Small interfaces (weak coupling)
   Untuk modul-modul yang berkomunikasi, diusahakan informasi yang dipertukarkan pada saat komunikasi adalah sesedikit mungkin sehingga mengurangi ketergantungan antar modul.
4. Explicit interface
   Kapan saja modul X dan Y berkomunikasi maka komunikasi ini harus dari teks X atau Y atau keduanya
5. Information hiding
   Pemrogram harus merancang modul dengan sekelompok fitur pada suatu modul tampak pada modul lain, sedangkan fitur lainnya diusahakan tersembunyi dari modul lain. Modul lain hanya berhubungan dengan modul lewat deskripsi pada fitur yang terlihat tersebut

PRINSIP MODULARITAS

Terdapat juga lima prinsip modularitas, yakni:

1. The Linguistic Modular Units principle

Modul harus merupakan unit sintaks pada bahasa pemrograman yang digunakan. Prinsip ini umumnya dilanggar karena itu pengembang terpaksa harus melakukan translasi atau restrukturisasi terhadap model rancangan yang diperolehnya.

2. The Self-Documentation Principle

Perancang modul harus membuat semua informasi mengenai modul yang berkaitan terdapat pada modul tersebut. Dokumentasi internal ini sangat penting untuk proses pengembangan dan pemeliharaan perangkat lunak

3. The Uniform Access Principle

Semua layanan modul seharusnya tersedia melalui notasi yang seragam tanpa memperhatikan pengimplementasian layanan tersebut apakah untuk keperluan penyimpanan atau komputasi.

4. The Open-Closed Principle

Modul harus bersifat terbuka dalam artian terbuka untuk dikembangkan serta bersifat tertutup dalam artian komunikasi antar modul hanya melalui antarmuka yang telah ditetapkan mekanismenya.

5. The Single Choice Principle

Kapan saja program harus mendukung beberapa alternatif, satu dan hanya satu modul pada program yang mengetahui daftar lengkap dari yang dimilikinya.

KRITERIA MODUL YANG BAIK

Beberapa kriteria dari modul yang baik antara lain:

1. Kohesif 5

Modul dikatakan kohesif jika fungsionalitasnya terdefinisi dan terfokus dengan baik. Kohesi mengacu pada derajat elemen-elemen modul yang saling berhubungan. Modul kohesif melakukan satu tugas tunggal pada suatu prosedur program yang memerlukan sedikit interaksi dengan prosedur yang sedang dilakukan pada bagian lain program.

Modul yang melakukan serangkaian tugas yang saling berhubungan secara lepas disebut sebagai kohesif koinsidental. Modul yang melakukan tugas yang berhubungan secara logis disebut kohesif secaralogis. Bila modul berisi tugas-tugas yang dieksekusi dalam jangka waktu sama, maka modul- modul tersebut disebut kohesif temporal. Bila elemen pemrosesan dari suatu modul dihubungkan dan dieksekusi dalam suatu urutan yang spesifik, maka akan muncul kohesi prosedural. Dan bila semua elemen pemrosesan berkonsentrasi pada satu area pada suatu struktur data, maka terjadi kohesi komunikasional

2. Loosely coupled

Coupling mengacu kepada derajat modul-modul saling berkomunikasi. Modul-modul harus seminimal mungkin berkomunikasi dengan modulmodul lain. Maka dari itu nilai derajat coupling harus sekecil mungkin

3. Enkapsulasi

Modul harus memenuhi persyaratan information hiding. Atribut dari modul seharusnya tidak secara langsung tersedia untuk modul-modul lain. Atribut-atribut modul hanya tersedia ke modul-modul lain melalui antarmuka yang telah ditetapkan. Enkapsulasi

mengimplikasikan pemahaman implementasi modul tertentu tidak dibutuhkan bagi pemakai modul sehingga tidak perlu mengetahui detail dankeseluruhan isi modul

4. Reuseability

Merupakan sasaran strategis rekayasa perangkat lunak dan dapat meningkatkan produktivitas pengembangan perangkat lunak. Implikasi dari reuseability adalah fungsionalitas modul harus segeneral dan seluas mungkin sehingga dapat digunakan oleh modul lain dan dapat mengurangi waktu dan biaya yang dikeluarkan.

ABSTRAKSI DATA

Abstraksi data merupakan suatu cara untuk menggambarkan data dengan memisahkannya dari implementasinya. Salah satu jenis abstraksi data adalah tipe data dan juga ADT (Abstract Data Type). Dengan abstraksi, seorang pemrogram tidak memperdulikan bagaimana data itu diimplementasikan,
contohnya tipe data int merupakan abstraksi dari sekumpulan bit di memori sebagai bilangan bulat. Tipe data merupakan sekumpulan nilai dan operasi yang diasosiasikan pada nilai-nilai itu. Sedangkan ADT mendeklarasikan sekumpulan nilai, operasi pada nilai, dan aksioma-aksioma yang senantiasa dipenuhi oleh operasioperasi tersebut. ADT tidak mendefinisikan cara nilai tersebut diimplementasikan sehingga mungkin terdapat beberapa implementasi berbeda untuk ADT yang sama.

CIRI-CIRI DARI ADT ADALAH:

Berisi struktur data dan operasi-operasi terhadap struktur data tersebut:

1. Menyediakan pengkapsulan
2. Menyediakan information hiding

3. Menyediakan abstraksi
4. Tidak menspesifikasikan implementasi struktur data
5. Menspesifikasikan perilaku dari ADT

KEGUNAAN ADT ANTARA LAIN:

ADT menyediakan dasar untuk modularitas perangkat lunak Mengidentifikasikan setiap modul dengan implementasi ADT, yaitu deskripsi sekumpulan objek dengan antarmuka bersama Antarmuka didefinisikan oleh sekumpulan operasi yang dibatasi oleh properti-properti yang abstrak masing operasi diimplementasikan menggunakan satu representasi dari ADT TERDAPAT TIGA KOMPONEN DALAM IMPLEMENTASI ADT, YAKNI: Spesifikasi ADT berisi fungsi-fungsi, aksioma-aksioma, dan prakondisiprakondisi. Pemilihan representasi bagi ADT Sekumpulan subprogram, masing-masing mengimplementasikan salah satu fungsi pada spesifikasi ADT yang beroperasi pada representasi yang telah dipilih

ANALISIS STATIK

Analisis statik merupakan proses menganalisis kode program yang dilakukan tanpa mengeksekusi kode program tersebut, berbeda dengan analisis dinamis di mana kode program dianalisis dengan mengeksekusi kode programnya. Terdapat beberapa alasan melakukan analisis statik antara lain: Analisis statik dapat dilakukan pada tahap awal pengkodean, tidak perlu menunggu sampai kode selesai dibuat Beberapa jenis kesalahan sulit untuk ditemukan melalui proses pengujian, misalkan kesalahan yang berkaitan dengan timing Proses pengujian dan analisis pada dasarnya bersifat komplemen, saling melengkapi satu sama lain .Metode formal digunakan untuk melakukan analisis statik dengan menggunakan serangkaian metode matematis. Teknik matematika yang digunakan antara lain semantik detonasional, semantik aksiomatik, semantic operasional, dan interpretasi abstrak. Salah satu implementasi dari teknik analisis statik adalah Data Flow Analysis

DATA FLOW ANALYSIS

Data flow analysis merupakan sebuah teknik untuk memperoleh informasi tentang sekumpulan nilai yang mungkin dihitung pada bagian tertentu pada program. Sebuah program Control Flow Graph (CFG) digunakan untuk menentukan bagian dari program di mana ketika nilai tertentu diberikan kepada variabel, maka kemungkinan terjadi propagasi. Informasi yang diperoleh seringkali digunakan oleh compiler untuk proses optimasi program.

Cara yang mudah untuk melakukan Data Flow Analysis adalah dengan menyediakan persamaan data flow untuk setiap node pada CFG dan menyelesaikannya dengan cara menghitung output dari input secara berulang pada setiap node secara lokal hingga keseluruhan sistem stabil.7 Setelah menyelesaikan persamaan ini, entri/exit state dari blok dapat digunakan untuk menjalankan properti program pada batasan blok.

Persamaan ini kemudian diselesaikan dengan cara iteratif hingga keseluruhan blok terselesaikan persamaannya. Ada dua pendekatan dari Data Flow Analysis, yakni:

Forward Analysis disebut juga sebagai Available Variable Analysis

Sebuah definisi disebut berguna jika dapat mempengaruhi komputasi pada lokasi yang bersangkutan Variabel yang tidak diinisialisasi mengindikasikan kesalahan.

Contoh penggunaannya:

1: if b==4 then 2: a = 5;
3: else
4: a = 3;
5: endif
6:
7: if a < 4 then

Definisi variabel a pada line 7 adalah sekumpulan nilai a=5 pada line 2 dan a=3 pada line 4

Backward Analysis disebut juga sebagai Like Variable Analysis Sebuah variabel disebut hidup jika nilainya saat ini digunakan untuk proses berikutnyaDead assignment mungkin mengindikasikan kesalahan Contoh penggunaannya:

```
// out: {} b1: a = 3; b = 5;
d = 4;
if a > b then

// in: {a,b,d}
// out: {a,b} b2: c = a + b; d = 2;
// in: {b,d}
// out: {b,d} b3: endif
c = 4;
return b * d + c;
// in:{}
```

TEKNIK IMPLEMENTASI PERANGKAT LUNAK (2)

Membuat, instalasi dan menguji jaringan

Terdapat berbagai jenis jaringan komputer, salah satunya LAN. Di sini kamu akan mempelajari cara membuat LAN sendiri. Walau sekarang WiFi lebih banyak dipakai, tapi tidak membuat LAN menjadi ditinggalkan. Instalasi LAN sendiri ditujukan untuk berbagai kebutuhan, di antaranya
sharing data antar komputer, mendapatkan kecepatan internet tinggi, hingga untuk ngegame, dan masih banyak lagi tentunya.
Alat Untuk Membuat Jaringan LAN:Seperti di judul, Pinhome akan menjelaskan langkah-langkah untuk membuat jaringan LAN. Namun sebelumnya kita butuh alat-alat tertentu, yaitu:

1. Kabel LAN

Pertama kabel LAN. Kabel ini ciri khasnya berwarna biru dengan konektor RJ-45, berfungsi sebagai media utama menyambungkan komputer ke jaringan. Kabel LAN bisa dibeli dengan murah di toko-toko komputer terdekat. Untuk panjang 1 sampai 2 meternya hanya sekitar Rp 20 ribuan.

Gambar 7. 1 Kabel LAN

2. Switch

Kalau kamu ingin membangun jaringan LAN dengan jumlah komputer yang terkoneksi lebih dari 1, maka wajib menggunakan switch. Alasannya, karena setiap komputer biasanya hanya memiliki 1 port untuk jaringan LAN. Nah, fungsi switch ini untuk membuat sebuah gerbang yang menghubungkan banyak komputer kedalam satu jaringan. Kamu pasti bisa paham hanya dengan melihat bentuk switchnya bukan?

Gambar 7. 2 Switch

3. Ethernet Card (Optional)

Untuk pengguna komputer yang tidak memiliki port LAN atau mungkin rusak, bisa coba untuk memakai Ethernet Card. Ethernet Card ini nantinya akan disambungkan ke slot PCI yang ada di motherboard komputer.

Gambar 7. 3 Ethernet Card

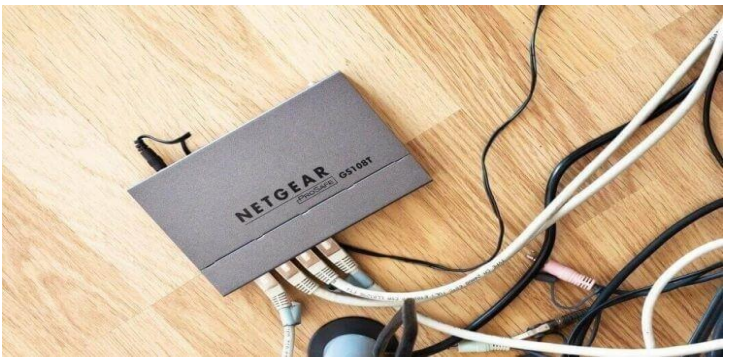
4. USB LAN (Optional)

Slot PCI komputer kamu sudah penuh? Solusinya bisa memakai USB LAN. Secara fungsi samasaja, yakni untuk membuat port LAN namun dengan media USB. Kamu bisa membelinya di toko-toko komputer terdekat.

Gambar 7. 4 USB LAN

5. Modem (Optional)

Karena hanya menyambungkan komputer kedalam suatu jaringan lokal, maka tidak tersedia akses internet pada LAN. Karena itu kita membutuhkan modem terlebih dahulu. Modem yang direkomendasikan bisa berbentuk router. Pada dasarnya, untuk membuat jaringan LAN kita hanya memerlukan kabelnya. Berikut ini cara membuat jaringan LAN sendiri.



Gambar 7. 5 Modem

Cara Membuat Jaringan LAN

Cara membuat jaringan LAN itu mudah, yang sulit itu adalah konfigurasi awalnya. Karena jika salah, komputer tidak akan bisa terhubung. Berikut ini langkah-langkah membuat jaringan LAN, bisa untuk Windows 7, Windows 8, dan Windows 10.

1. Hubungkan Terlebih Dahulu Komputer


Gambar 7. 6 Port Laptop untuk LAN

Langkah pertama, silakan hubungkan terlebih dahulu komputer (atau laptop) melalui port menggunakan kabel LAN.

Caranya:

Kamu bisa sambung dua komputer secara langsung dengan kabel LAN (sederhana). Kemudian, sambung dua atau lebih komputer

langsung ke switch atau router.

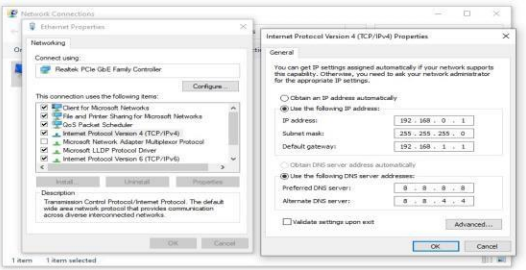
Setelah itu kamu bisa menggunakan topologi jaringan. Ciri LAN yang sudah aktif, biasanya akan terdapat lampu indikator menyala berwarna kuning atau merah. Lampu tersebut berkedip, menandakan bahwa LAN sudah berjalan.

Nonaktifkan Pengaturan Jaringan Firewall

Gambar 7. 7 Nonatifkan Firewall

Firewall adalah sebuah fitur pada Windows yang berfungsi sebagai keamanan
saat user sedang browsing di internet dan bisa mempercepat internet. Walau bermanfaat, Firewall ini bisa membuat jaringan LAN yang kita buat justru jadi tidak bisa terhubung (terblokir). Karena itu, direkomendasikan untuk menonaktifkannya terlebih dahulu. Caranya seperti ini:
Silakan buka terlebih dahulu Control Panel Windows. Buka menu System and Security.
Kemudian klik bagian Windows Defender Firewall.
Lalu pada bagian menu kiri, pilih Turn Windows Defender Firewall to on or off. Kemudian setting Private network settings dan Public network settings menjadi off. Kalau sudah klik Ok. Sekarang jaringan LAN sudah siap dibuat.

Atur Konfigurasi Jaringan LAN di Komputer Pertama



Gambar 7. 8 Konfigurasi Jaringan LAN di Laptop

Membuat jaringan LAN bisa dilakukan di komputer dengan versi Windows apa saja. Namun disarankan, minimal pakai Windows 7 atau versi diatasnya, untuk mendapatkan fitur keamanan yang bagus.

Pertama setting dulu IP Address, berikut caranya:

Pertama buka Control Panel di Windows. Kemudian masuk ke bagian Network and Internet. Buka lagi menu Network and Sharing Center.

Klik pada bagian Change adapter settings.

Pilih koneksi Ethernet > Klik kanan lalu pilih Properties.

Pilih pada bagian Internet Protocol (TCP/IPv4) lalu klik tombol Properties. Isi IP Address secara manual menjadi:

IP Address 192.168.1.1

Subnet mask 255.255.255.0

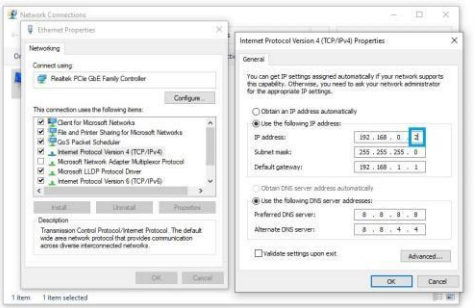
Default Gateway Isi sesuai dengan IP modem

Untuk DNS bisa dikosongkan. Namun kalau ingin diisi juga bisa, menggunakan DNS Google.

Preferred DNS Server 8,8.8.8

Alternate DNS Server 8.8.4.4

Pengaturan LAN tersebut di Windows 7, Windows 8, Windows 10 sama langkah-langkahnya.



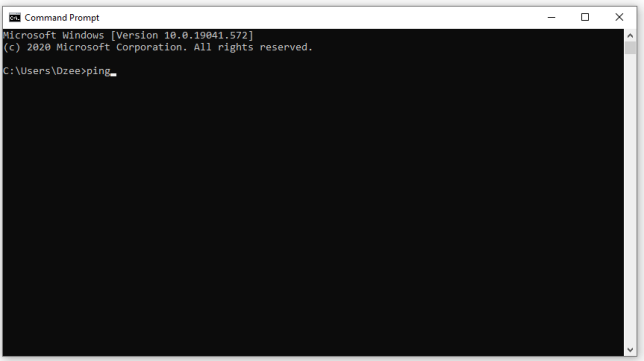
Cara Mengatur Konfigurasi LAN untuk Komputer Kedua

Setelah komputer pertama selesai diatur, langkah selanjutnya adalah mengaktifkan konfigurasi LAN pada komputer 2 dan komputer-komputer lainnya. Caranya kurang lebih sama, yakni melalui *properties* di Internet Protocol (TCP/IPv4) Control Panel. Bedanya ada pengaturan IP, dimana komputer seterusnya harus memiliki akhir angka yang berbeda. Contohnya begini:

Komputer 1: 192.168.1.**1**
Komputer 2: 192.168.1.**2**
Komputer 3: 192.168.1.**3**
Komputer 4: 192.168.1.**4**

Sedangkan untuk pengaturan lainnya, seperti *subnet mask*, *default gateway* maupun pengaturan DNS bisa disamakan. Kecuali kalau kamu hanya menyambungkan dua komputer saja. Untuk bagian subnet mask harus dibuat begini: **Subnet Mask komputer 1** : 192.168.1.2 **Subnet Mask komputer 2** : 192.168.1.1 Dan seterusnya

Mengetes LAN



Gambar 7. 9 Testing LAN

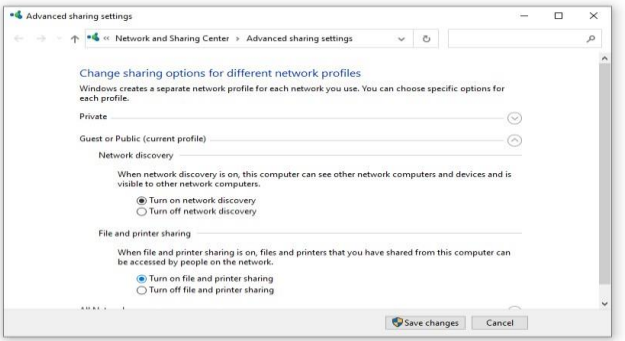
Kalau kamu sudah mengikuti langkah-langkah diatas dengan benar, maka otomatis jaringan LAN bisa langsung dipakai. Kamu bisa mengecek koneksi LAN-nya sudah terhubung atau belum, menggunakan cara berikut:

1. Komputer 1:
2. Klik tombol start Windows. Kemudian buka CMD.
3. Setelah itu ketik “ping 192.168.1.2” lalu tekan Enter.
4. Komputer 2:
5. Klik tombol start Windows. Kemudian buka CMD.
6. Setelah itu ketik “ping 192.168.1.1” lalu tekan Enter.

Ciri jaringan sudah terhubung, akan muncul pesan reply. Jika gagal, nanti akan muncul
pesan timed out. Kalau kamu mendapatinya timed out silakan dicoba atur-atur lagi. Langkah ini sering juga disebut dengan istilah ping.

Solusi Jika LAN Tanda Silang
Biasanya setelah kita menghubungkan laptop atau komputer via

LAN, bisa langsung terhubung, bisa juga muncul tanda silang / silang merah. Untuk ini solusinya tidak sulit.Periksa pengaturan IP pada LAN. Biasanya belum benar. Kalau menghubungkan banyak komputer, periksa topologinya. Pastikan Firewall sudah mati dengan benar. Restart semua komputer kemudian coba lagi. Cara Sharing Data Menggunakan Kabel LAN



Gambar 7. 10 Sharing Data

Jaringan LAN sudah tersambung? Maka kamu bisa mulai menggunakan fitur-fitur yang terdapat pada jaringan. Salah satu yang mungkin paling dibutuhkan ialah sharing data. Dengan sharing data ini kamu bisa berbagi data pribadi antara 1 komputer dengan komputer lainnya, dengan kecepatan transfer yang sangat cepat.

Sebelum mengaturnya, kamu harus memastikan dua hal berikut:

Pastikan kamu sudah menonaktifkan Firewall.

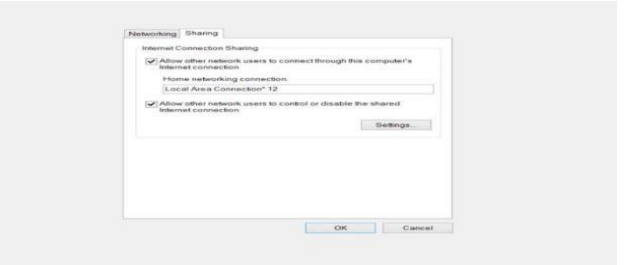
Perintah ping yang kamu lakukan di CMD sudah mendapatkan pesan reply. Konfigurasi yang dilakukan sudah benar, terutama pada bagian subnet mask. Untuk langkah-langkahnya begini:

Buka Control Panel.

Masuk ke menu Network and Internet lalu buka Network and Sharing Center. Klik pada Connections yang sedang terhubung.

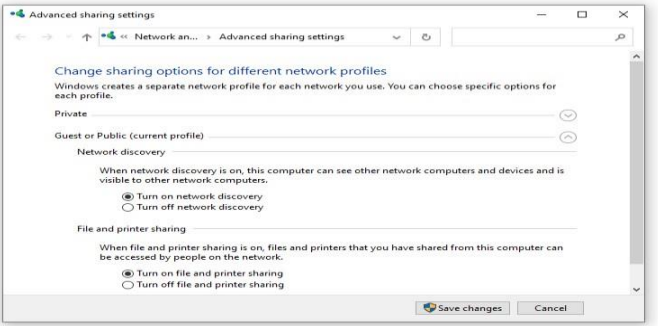
Klik Properties > pilih tab Sharing > kemudian centang Allow others network users to connect through this computer’s Internet

connection.

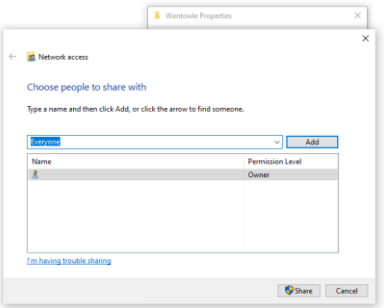


Kalau sudah, lanjut klik Change Advanced Sharing Settings yang ada pada menu bagian kiri, kemudian aktifkan fitur-fitur berikut:

a. Turn on network discovery.
b. Turn on file printer sharing.
c. Pada Public Folder aktifkan Turn on sharing. Nonaktifkan Password protected sharing.



Pengaturan tersebut harus kamu lakukan di semua komputer yang terhubung ke LAN. Dan selanjutnya, kamu buka komputer yang ingin dibagikan datanya, lalu ikuti langkah-langkah ini: Klik kanan folder atau file yang ingin dibagikan > Lalu pilih Properties.
Masuk ke tab Sharing lalu klik Share. Pada menu dropdown > Pilih Everyone. Aktifkan opsi Read / Write.
Lalu klik Share.



Nah, kamu bisa membuka folder tersebut dengan memasukkan IP komputer yang dibagikan melalui menu Run. Caranya klik tombol Windows + R > Masukkan IP komputernya > Lalu klik Ok.

Sekilas Tentang LAN

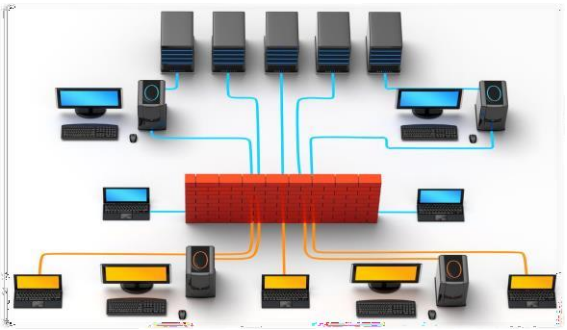
LAN adalah singkatan dari Local Area Network, yang berarti Jaringan Area Lokal. Maksudnya adalah sebuah jaringan komputer yang dibangun di suatu tempat untuk kebutuhan akses koneksi. Misalnya di suatu kantor, terdapat satu modem yang menyediakan akses internet,sedangkan clientnya ada 20. Untuk membuat seluruh client tersebut bisa mengakses internet, dibangunlah jaringan LAN. Ada beberapa keuntungan menggunakan jaringan LAN, yaitu:

a. **Hemat biaya**, karena akses internet dari provider yang dibutuhkan hanya satu, dan kita bisa membagikannya ke banyak users.
b. **Mudah membuatnya**, karena tidak memerlukan banyak alat.
c. **Bisa untuk transfer file**, misalnya antar komputer. Tentu ini bisa menghemat waktu, ketimbang menggunakan perangkat eksternal.

Bisa menghubungkan ke perangkat selain komputer, seperti printer. Banyak loh printer yang bisa menggunakan jaringan LAN untuk dioperasikan.

Kesimpulan

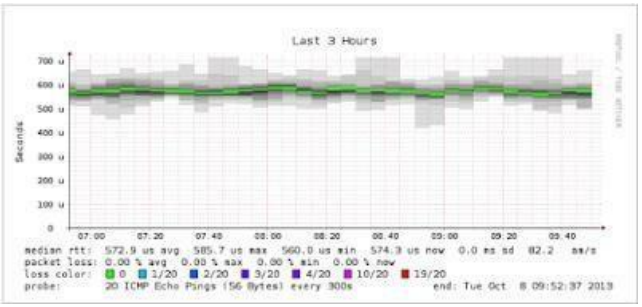
Demikianlah sedikit ulasan tentang cara membuat jaringan LAN sendiri. Setting awalnya mungkin memang relatif agak ribet, tapi sifatnya sesekali saja. Alernatif membuat jaringan LAN sendiri ini bisa membantumu untuk menghemat budget dan memudahkan PC atau laptop untuk terhubung ke perangkat lain. Sekian ulasan pinhome kali ini, semoga bermanfaat!.



Pengujian Jaringan Komputer

Pengujian Latency

Latency berhubungan dengan seberapa banyak waktu yang dibutuhkan untuk mengirim pesan dari ujung jaringan ke ujung yang lain. Pengukuran Latency ini menggunakan aplikasi Smokeping. Pengukuran dimaksudkan untuk melihat banyak pengaksesan web tersebut. Pengukuran latency dilakukan selama satu hari.



Pengujian Benchmark

Teknik pengujian dengan menggunakan suatu nilai standar. Suatu program atau performa pekerjaan yang melakukan perbandingan kemampuan dari berbagai kinerja dari beberapa peralatan dengan

tujuan untuk meningkatkan kualitas pada produk yang baru. Pengujian dilakukan dengan cara membandingkan produk-produk perangkat lunak maupun perangkat keras dengan percobaan yang sama.

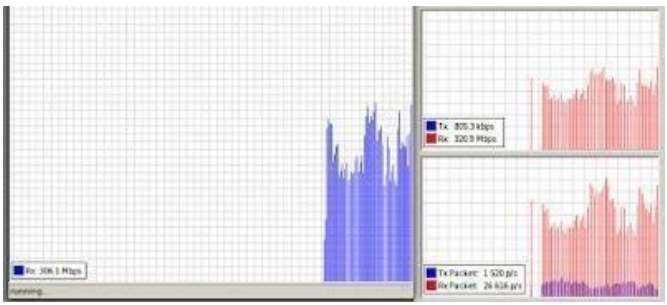
Bandwidth adalah perhitungan konsumsi transfer data telekomunikasi yang dihitung dalam satuan bit per detik atau yang biasa disingkat bps yang terjadi antara komputer server dan komputer client dalam waktu tertentu dalam sebuah jaringan komputer. Pengukuran Bandwidth ini menggunakan aplikasi NTOP. Pengukuran bandwidth dilakukan agar dapat dilihat jumlah data yang ditransfer dalam pengaksesan instance yang diakses oleh user.

Host	Domain	Data		TCP	UDP	ICMP
www.el		90.6 MBytes	62.2 %	84.3 MBytes	77.4 KBytes	6.2 MBytes
192.168.60.1		44.4 MBytes	30.5 %	44.2 MBytes	0	118.3 KBytes
192.168.60.200		10.5 MBytes	7.2 %	7.9 MBytes	0	2.5 MBytes
172.19.1.1		112.1 KBytes	0.1 %	4.9 KBytes	35.4 KBytes	0
::		53.3 KBytes	0.0 %	0	0	0
ip6-allnodes		37.1 KBytes	0.0 %	0	0	0
169.254.169.254		35.4 KBytes	0.0 %	0	35.4 KBytes	0
all-systems.mcast.net		19.9 KBytes	0.0 %	0	0	0
laa (locally assigned address):f5:07:f3		16.2 KBytes	0.0 %	0	0	0

Bandwidth Test Menggunakan Mikrotik

Disamping fungsi utama manajemen jaringan, Router Mikrotik juga mempunyai tool yang bisa digunakan untuk mengetahui seberapa besar traffic yang bisa dilewatkan pada sebuah link atau jalur koneksi. Tool yang dimaksud adalah Btest Server dan Bandwidth Test. Bisa diakses

pada menu /tool. Mikrotik akan men-generate traffic yang kemudian akan dikirimkan ke perangkat lain melalui sebuah jalur koneksi. Proses ini biasa disebut dengan Bandwidth test. Sebuah proses Bandwidth test terdiri dari Bandwidth test server dan Bandwidth test client. Semua versi RouterOS Mikrotik bisa digunakan sebagai Bandwidth Test server maupun Bandwidth test client.



Membuat struktur basisdata dan menguji basis data

Membuat Struktur Database

Panduan ini mencakup beberapa konsep utama dalam arsitektur data dan praktik terbaik untuk merancang struktur data JSON dalam Firebase Realtime Database. Membuat database yang terstruktur dengan baik membutuhkan perencanaan yang matang. Yang terpenting, Anda harus merencanakan bagaimana data akan disimpan dan diambil kembali untuk membuat proses tersebut menjadi semudah mungkin. Metode perancangan struktur data: dengan hierarki JSON. Semua data Firebase Realtime Database disimpan sebagai objek JSON. Anda dapat menganggap database sebagai hierarki JSON yang dihosting di cloud. Tidak seperti database SQL, database tersebut tidak memiliki tabel atau catatan. Ketika Anda menambahkan data ke hierarki JSON, data tersebut akan menjadi node di struktur JSON yang ada dan memiliki kunci terkait. Anda dapat membuat kunci sendiri, misalnya dengan ID pengguna atau nama semantik. Kunci juga dapat dibuatkan untuk Anda menggunakan push().

Jika Anda membuat kunci sendiri, kunci tersebut harus berenkode UTF-8, berukuran maksimum 768 byte, dan tidak boleh berisi karakter ., $, #, [,], /, atau karakter kontrol ASCII 0-31 atau 127. Anda juga tidak dapat menggunakan karakter kontrol ASCII dalam nilai-nilai itu sendiri.

Misalnya, bayangkan sebuah aplikasi chat yang memungkinkan pengguna untuk menyimpan profil dasar dan daftar kontak. Profil pengguna biasanya terletak pada suatu jalur, misalnya /users/$uid. Pengguna alovelace mungkin memiliki entri database yang terlihat seperti ini:

```
{
"users": {
"alovelace": {
"name": "Ada Lovelace", "contacts": { "ghopper": true },
},
"ghopper": { ... },
"eclarke": { ... }
}
}
```

Meskipun database ini menggunakan hierarki JSON, data yang tersimpan di database dapat digambarkan sebagai jenis native tertentu yang berkaitan dengan jenis JSON yang tersedia. Dengan begitu, Anda dapat menulis kode yang lebih mudah dikelola.

Praktik terbaik untuk struktur data Hindari data bertingkat

Karena Firebase Realtime Database memungkinkan data bertingkat hingga kedalaman 32 tingkat, Anda mungkin berpikir bahwa ini adalah struktur defaultnya. Namun, saat Anda mengambil data di sebuah lokasi dalam database Anda, Anda juga mengambil semua node turunannya. Selain itu, ketika Anda memberikan akses baca atau tulis kepada seseorang pada sebuah node di database Anda,

Anda juga memberinya akses ke semua data di bawah node tersebut. Oleh karena itu, dalam praktiknya, lebih baik buat struktur data Anda serata mungkin.

Untuk memberi gambaran mengapa data yang bertingkat itu buruk, perhatikan struktur multi-tingkat berikut:

```
{
// This is a poorly nested data architecture, because iterating the children
// of the "chats" node to get a list of conversation titles requires
// potentially downloading hundreds of megabytes of messages
"chats": {
"one": {
"title": "Historical Tech Pioneers", "messages": {
"m1": { "sender": "ghopper", "message": "Relay malfunction found. Cause: moth." },
"m2": { ... },

// a very long list of messages
}
},
"two": { ... }
}
}
```

Dengan desain bertingkat ini, iterasi data bisa bermasalah. Misalnya, untuk menampilkan daftar judul percakapan chat, seluruh hierarki chats harus didownload ke klien, termasuk semua anggota dan pesannya.

Ratakan struktur data

Jika data tersebut dibagi ke beberapa jalur, disebut juga dengan denormalisasi, data tersebut dapat didownload secara efisien dalam panggilan terpisah, sesuai dengan kebutuhan. Perhatikan struktur yang diratakan ini:

```
{
// Chats contains only meta info about each conversation
```

```
// stored under the chats's unique ID "chats": {
"one": {
"title": "Historical Tech Pioneers",
"lastMessage": "ghopper: Relay malfunction found. Cause: moth.",
"timestamp": 1459361875666
},
"two": { ... },
"three": { ... }
},

// Conversation members are easily accessible
// and stored by chat conversation ID "members": {
// we'll talk about indices like this below "one": {
"ghopper": true, "alovelace": true, "eclarke": true
},
"two": { ... },
"three": { ... }
},

// Messages are separate from data we may want to iterate quickly
// but still easily paginated and queried, and organized by chat

// conversation ID "messages": {
"one": {
"m1": {
"name": "eclarke",
"message": "The relay seems to be malfunctioning.", "timestamp":
1459361875337
},
"m2": { ... },
"m3": { ... }
},
"two": { ... },
"three": { ... }
}
```

}

Sekarang, iterasi daftar ruang chat dapat dilakukan cukup dengan mendownload beberapa byte untuk setiap percakapan. Dengan begitu, pengambilan metadata untuk menampilkan ruang chat di UI bisa dilakukan dengan cepat. Pesan dapat diambil secara terpisah dan ditampilkan saat pesan tersebut tiba, sehingga UI akan tetap cepat dan responsif.

Buat data yang dapat diskalakan

Ketika mem-build aplikasi, mendownload subkumpulan daftar sering kali merupakan keputusan yang lebih baik. Praktik ini biasa dilakukan, terutama jika daftar tersebut memuat ribuan catatan. Jika hubungan ini bersifat statis dan satu arah, Anda cukup menempatkan objek turunan di bawah induk. Terkadang, hubungan ini bersifat lebih dinamis, atau denormalisasi mungkin perlu dilakukan pada data ini. Biasanya, denormalisasi data dapat dilakukan menggunakan kueri untuk mengambil subkumpulan data, seperti yang dibahas dalam bagian Mengambil Data.

Namun, ini pun mungkin tidak cukup. Bayangkan, misalnya, hubungan dua arah antara pengguna dan grup. Pengguna dapat menjadi anggota sebuah grup, sementara grup memiliki daftar sejumlah pengguna. Keadaan menjadi rumit saat Anda harus menentukan grup mana saja yang diikuti pengguna. Tentu diperlukan cara yang praktis untuk membuat daftar grup yang diikuti pengguna dan hanya mengambil data untuk grup tersebut. Indeks grup dapat sangat membantu di sini:

```
// An index to track Ada's memberships
{
"users": {
"alovelace": {
"name": "Ada Lovelace",
// Index Ada's groups in her profile "groups": {
// the value here doesn't matter, just that the key exists
"techpioneers": true,
"womentechmakers": true
```

```
}
},
...
},
"groups": { "techpioneers": {
"name": "Historical Tech Pioneers", "members": {
"alovelace": true, "ghopper": true, "eclarke": true
}
},
...
}
}
```

Anda mungkin menyadari bahwa tindakan ini akan menduplikat beberapa data dengan cara menyimpan hubungan di catatan Ada maupun di grup. Sekarang alovelace diindeks dalam grup dan techpioneers tercantum pada profil Ada. Jadi, untuk menghapus Ada dari grup, update harus dilakukan di dua tempat.

Ini adalah hal yang harus dilakukan untuk hubungan dua arah. Dengan begitu, Anda dapat mengambil keanggotaan Ada dengan cepat dan efisien, bahkan ketika daftar pengguna atau grup diskalakan hingga jutaan, atau ketika aturan keamanan Realtime Database menghalangi akses ke beberapa catatan tersebut.

Pendekatan ini, yang membalikkan data dengan mencantumkan ID sebagai kunci dan menetapkan nilai ke benar (true) membuat proses pencarian kunci menjadi semudah membaca /users/$uid/groups/$group_id dan memeriksa apakah nilainya null. Indeks ini lebih cepat dan jauh lebih efisien daripada proses kueri atau memindai data.

Menyimpan data ke Realtime Database

Membaca dan Menulis Data di Web

(Opsional) Membuat prototipe dan melakukan pengujian dengan Firebase Local Emulator Suite

Sebelum membahas cara aplikasi Anda membaca dari dan menulis ke Realtime Database, kami akan memperkenalkan serangkaian alat yang dapat digunakan untuk membuat prototipe dan menguji fungsionalitas Realtime Database: Firebase Local Emulator Suite. Jika Anda sedang mencoba berbagai model data, mengoptimalkan aturan keamanan, atau berupaya menemukan cara yang paling hemat untuk berinteraksi dengan backend, kemampuan untuk bekerja secara lokal tanpa men-deploy layanan langsung dapat sangat bermanfaat.

Emulator Realtime Database adalah bagian dari Local Emulator Suite yang memungkinkan aplikasi berinteraksi dengan konfigurasi dan konten database yang diemulasi, serta secara opsional dengan resource project yang diemulasi (fungsi, database lain, dan aturan keamanan).

Anda hanya perlu beberapa langkah untuk menggunakan emulator Realtime Database: Menambahkan satu baris kode ke konfigurasi pengujian aplikasi untuk terhubung ke emulator. Menjalankan firebase emulators:start dari root direktori project lokal Anda.

Melakukan panggilan dari kode prototipe aplikasi Anda menggunakan SDK platform Realtime Database seperti biasa, atau menggunakan Realtime Database REST API.

Menulis data

Dokumen ini mencakup dasar-dasar pengambilan data, serta cara mengurutkan dan memfilter data Firebase.

Data Firebase diambil dengan menambahkan pemroses asinkron ke firebase.database.Reference. Pemroses dipicu sekali untuk status awal data, dan dipicu kembali setiap kali data berubah.

Catatan: Secara default, akses baca dan tulis pada database Anda dibatasi sehingga hanya pengguna terautentikasi yang dapat membaca atau menulis data. Untuk memulai tanpa menyiapkan Authentication, Anda dapat mengonfigurasi aturan untuk akses publik. Tindakan ini membuat database terbuka bagi siapa saja, termasuk orang yang tidak menggunakan aplikasi Anda. Oleh sebab itu, pastikan untuk membatasi database lagi saat menyiapkan autentikasi.

Operasi tulis dasar

Untuk operasi tulis dasar, Anda dapat menggunakan set() untuk menyimpan data ke referensi yang ditentukan, sehingga menggantikan data yang ada di jalur tersebut. Misalnya, aplikasi blogging sosial dapat menambahkan pengguna dengan set() seperti berikut:

```
import { getDatabase, ref, set } from "firebase/database";

function writeUserData(userId, name, email, imageUrl) { const db =
getDatabase();
set(ref(db, 'users/' + userId), { username: name,
email: email, profile_picture : imageUrl
});
}
```

Jika set() digunakan, data di lokasi yang ditentukan akan ditimpa, termasuk semua node turunannya.

Membaca data Memproses peristiwa nilai

Untuk membaca data di suatu jalur dan memproses perubahan, gunakan onValue() untuk mengamati peristiwa. Anda dapat menggunakan peristiwa ini untuk membaca snapshot statis konten di

jalur tertentu, sesuai keberadaan konten tersebut selama peristiwa terjadi. Metode ini terpicu satu kali ketika pemroses terpasang, dan terpicu lagi setiap kali terjadi perubahan pada data, termasuk pada setiap turunannya. Callback peristiwa mendapatkan snapshot yang berisi semua data di lokasi tersebut, termasuk data turunan. Jika tidak ada data, snapshot akan menampilkan false ketika exists() dipanggil, serta menampilkan null ketika val() dipanggil pada snapshot tersebut.

Penting: onValue() akan dipanggil setiap kali data diubah pada referensi database yang ditentukan, termasuk perubahan pada turunannya. Untuk membatasi ukuran snapshot, tambahkan pemroses hanya

pada tingkat terendah yang diperlukan untuk memantau perubahan. Misalnya, menambahkan pemroses ke root database tidak dianjurkan.

Contoh berikut menunjukkan aplikasi blogging sosial yang mengambil jumlah bintang suatu postingan dari database:

```
import { getDatabase, ref, onValue} from "firebase/database";

  const db = getDatabase();
  const starCountRef = ref(db, 'posts/' + postId + '/starCount');
  onValue(starCountRef, (snapshot) => {
  const data = snapshot.val(); updateStarCount(postElement, data);
  });
```

Pemroses akan menerima snapshot yang berisi data di lokasi yang ditentukan dalam database saat peristiwa terjadi. Anda dapat mengambil data dalam snapshot dengan metode val().

Membaca data sekali

Membaca data sekali dengan get()

SDK didesain untuk mengelola interaksi dengan server database, baik saat aplikasi Anda online maupun offline.

Biasanya, Anda harus menggunakan teknik peristiwa nilai yang dijelaskan di atas untuk membaca data agar mendapatkan notifikasi terkait pembaruan data dari backend. Teknik pemroses mengurangi penggunaan dan penagihan Anda, serta dioptimalkan untuk memberikan pengalaman terbaik kepada pengguna saat mereka online dan offline.

Jika hanya memerlukan data satu kali, Anda dapat menggunakan get() untuk mendapatkan snapshot data dari database. Jika karena alasan apa pun get() tidak dapat menampilkan nilai server, klien akan menyelidiki cache penyimpanan lokal dan menampilkan error jika nilainya masih belum ditemukan.

Penggunaan get() yang tidak perlu dapat meningkatkan penggunaan bandwidth dan menyebabkan penurunan performa. Ini dapat dicegah menggunakan pemroses realtime seperti yang ditunjukkan di atas.

```
import { getDatabase, ref, child, get } from "firebase/database";
const dbRef = ref(getDatabase());
get(child(dbRef, `users/${userId}`)).then((snapshot) => { if (snapshot.exists()) {

  console.log(snapshot.val());
  } else {
  console.log("No data available");
  }
  }).catch((error) => { console.error(error);
  });
```

Membaca data sekali dengan observer

Dalam beberapa kasus, Anda mungkin menginginkan nilai dari cache lokal langsung ditampilkan, daripada memeriksa nilai yang diperbarui di server. Dalam kasus tersebut, Anda dapat

menggunakan once() untuk langsung mendapatkan data dari cache disk lokal.

Cara ini berguna untuk data yang hanya perlu dimuat sekali, dan tidak diharapkan untuk sering berubah atau memerlukan pemrosesan aktif. Misalnya, aplikasi blogging pada contoh sebelumnya menggunakan metode ini untuk memuat profil pengguna ketika mulai membuat postingan baru:

```
import { getDatabase, ref, onValue } from "firebase/database";
import { getAuth } from "firebase/auth";
const db = getDatabase(); const auth = getAuth();const userId =
auth.currentUser.uid;
return onValue(ref(db, '/users/' + userId), (snapshot) => { const
username = (snapshot.val() && snapshot.val().username) ||
'Anonymous';
// ...
}, {
onlyOnce: true
});
```

Memperbarui atau menghapus data Memperbarui kolom tertentu Untuk menulis secara simultan ke turunan tertentu sebuah node tanpa menimpa node turunan yang lain, gunakan metode update(). Saat memanggil update(), Anda dapat memperbarui nilai turunan pada level yang lebih rendah dengan menetapkan jalur untuk kunci. Jika data disimpan dalam beberapa lokasi agar dapat melakukan penskalaan yang lebih baik, Anda dapat memperbarui semua instance data tersebut menggunakan fan- out data.
Misalnya, aplikasi blogging sosial dapat membuat postingan dan mengupdatenya secara bersamaan ke feed aktivitas terbaru dan feed aktivitas pengguna pengirim postingan menggunakan kode seperti ini:

```
function writeNewPost(uid, username, picture, title, body) {const db
= getDatabase();
// A post entry. const postData = {author: username, uid: uid,
body: body, title: title, starCount: 0, authorPic: picture


};

// Get a key for a new Post.
const newPostKey = push(child(ref(db), 'posts')).key;
// Write the new post's data simultaneously in the posts list andthe
user's post list.
const updates = {};
updates['/posts/' + newPostKey] = postData; updates['/user-posts/'
+ uid + '/' + newPostKey] = postData;return update(ref(db), updates);
}
```

Contoh ini menggunakan push() untuk membuat postingan dalam node yang berisi postingan bagi semua pengguna di /posts/$postid dan sekaligus mengambil kunci. Selanjutnya, kunci tersebut dapat digunakan untuk membuat entri kedua di postingan pengguna pada /user- posts/$userid/$postid.

Dengan menggunakan jalur tersebut, Anda dapat menjalankan pembaruan simultan ke beberapa lokasi di hierarki JSON dengan satu panggilan ke update(), seperti yang digunakan pada contoh ini untuk membuat postingan baru di kedua lokasi. Pembaruan simultan yang dilakukan dengan cara ini bersifat atomik: semuanya akan berhasil atau semuanya akan gagal.

Menambahkan Callback Penyelesaian

Jika Anda ingin tahu kapan data di-commit, Anda bisa menambahkan callback penyelesaian. set() dan update() menerapkan callback penyelesaian opsional yang akan dipanggil ketika operasi tulis telah di-commit ke database. Jika panggilan tidak berhasil, objek error akan diteruskan ke callback untuk

menunjukkan penyebab kegagalan.

```
import { getDatabase, ref, set } from "firebase/database";const db =
getDatabase(); set(ref(db, 'users/' + userId), { username: name,
email: email,
profile_picture : imageUrl
})
.then(() => {
// Data saved successfully!
})
.catch((error) => {
// The write failed...
});
```

Menghapus data

Cara termudah untuk menghapus data adalah dengan memanggil remove() pada referensi ke lokasi data tersebut.

Penghapusan juga dapat dilakukan dengan menentukan null sebagai nilai untuk operasi tulis lainnya, seperti set() atau update(). Teknik ini dapat digunakan dengan update() untuk menghapus beberapa turunan dengan satu panggilan API.

Menerima Promise

Untuk mengetahui kapan data di-commit ke server Firebase Realtime Database, Anda dapat menggunakan Promise. set() dan update() dapat menampilkan Promise yang dapat Anda gunakan untuk mengetahui kapan penulisan di-commit ke database.

Melepas pemroses

Callback akan dihapus dengan memanggil metode off() pada referensi database Firebase.

Anda dapat menghapus sebuah pemroses dengan meneruskannya sebagai parameter ke off(). Memanggil off() pada lokasi tanpa argumen akan menghapus semua pemroses yang ada di lokasi tersebut.

Memanggil off() pada pemroses induk tidak akan otomatis menghapus pemroses yang terdaftar pada node turunannya. off() juga harus dipanggil pada pemroses turunan mana pun untuk menghapus callback.

Menyimpan data sebagai transaksi

Ketika menangani data yang bisa rusak karena perubahan serentak, seperti penghitung pertambahan inkremental, Anda dapat menggunakan operasi transaksi. Anda dapat memberi operasi ini fungsi pembaruan dan callback penyelesaian opsional. Fungsi pembaruan mengambil status data saat ini sebagai argumen, dan akan menampilkan status baru yang ingin Anda tulis. Jika ada klien lain yang melakukan penulisan ke lokasi sebelum nilai baru Anda berhasil ditulis, fungsi pembaruan Anda akan dipanggil lagi dengan nilai saat ini yang baru, dan penulisan akan dicoba ulang.

Misalnya, pada contoh aplikasi blogging sosial, Anda dapat mengizinkan pengguna memberi atau menghapus bintang pada postingan, serta memantau jumlah bintang yang telah diterima suatu postingan dengan cara berikut ini:

```
import { getDatabase, ref, runTransaction } from"firebase/database";
function toggleStar(uid) { const db = getDatabase();const postRef = ref(db, '/posts/foo-bar-123'); runTransaction(postRef, (post) => { if (post) {
if (post.stars && post.stars[uid]) { post.starCount--;post.stars[uid] = null;
} else { post.starCount++; if (!post.stars) {post.stars = {};
}
post.stars[uid] = true;
}
}
```

```
return post;
});
}
```

Penggunaan transaksi dapat mencegah kesalahan penghitungan jumlah bintang jika beberapa pengguna memberi bintang pada postingan yang sama secara bersamaan, atau jika klien memiliki data yang sudah usang. Jika transaksi ditolak, server akan menampilkan nilai saat ini ke klien, yang akan mengulangi transaksi tersebut dengan nilai yang telah diperbarui. Proses ini akan berulang sampai transaksi diterima atau Anda membatalkan transaksi.

Catatan: Karena dipanggil berkali-kali, fungsi pembaruan harus dapat menangani data null. Meskipun ada data dalam database jarak jauh Anda, data tersebut mungkin tidak ditempatkan di cache lokal saat fungsi transaksi dijalankan, sehingga null dihasilkan sebagai nilai awal.

Pertambahan inkremental atomik sisi server.

Dalam kasus penggunaan di atas, kita menulis dua nilai ke database: ID pengguna yang memberi/menghapus bintang pada postingan, dan pertambahan inkremental jumlah bintang. Jika sudah mengetahui bahwa pengguna memberi bintang pada postingan, kita dapat menggunakan operasi pertambahan inkremental atomik, bukan transaksi.

```
function addStar(uid, key) { const updates = {};
updates[`posts/${key}/stars/${uid}`] = true;
updates[`posts/${key}/starCount`] =
firebase.database.ServerValue.increment(1);
    updates[`user-posts/${key}/stars/${uid}`] = true;
    updates[`user-posts/${key}/starCount`] =
firebase.database.ServerValue.increment(1);
```

```
firebase.database().ref().update(updates);
}
```

Kode ini tidak menggunakan operasi transaksi, sehingga tidak otomatis dijalankan ulang jika ada pembaruan yang bertentangan. Namun, karena operasi pertambahan inkremental terjadi langsung di server database, tidak ada kemungkinan konflik.

Jika ingin mendeteksi dan menolak konflik khusus aplikasi, misalnya pengguna memberi bintang pada postingan yang sebelumnya telah dibintanginya, Anda harus menulis aturan keamanan khusus untuk kasus penggunaan tersebut.

Menangani data secara offline

Jika koneksi jaringan klien terputus, aplikasi Anda akan tetap berfungsi dengan baik.

Setiap klien yang terhubung ke database Firebase menyimpan versi internalnya sendiri untuk setiap data aktif. Ketika ditulis, data akan dituliskan ke versi lokal ini terlebih dahulu. Selanjutnya, klien Firebase menyinkronkan data tersebut dengan server database jarak jauh, dan dengan klien lain berdasarkan "upaya terbaik".

Akibatnya, semua operasi tulis ke database akan langsung memicu peristiwa lokal, sebelum ada data yang dituliskan ke server. Ini berarti aplikasi Anda akan tetap responsif, apa pun kondisi konektivitas atau latensi jaringannya.

Setelah terhubung kembali ke jaringan, aplikasi Anda akan menerima kumpulan peristiwa yang sesuai, sehingga klien dapat menyinkronkannya dengan kondisi server saat ini, tanpa harus menulis kode khusus.

Catatan: API web Firebase Realtime Database tidak akan terus mempertahankan data secara offline di luar sesi. Agar penulisan tetap dipertahankan di server, halaman web tidak boleh ditutup sebelum data ditulis ke server.

DAFTAR PUSTAKA

1. Spillner, A., Linz, T., & Schaefer, H. (2021). *Software Testing Foundations*. Rocky Nook.
2. Jorgensen, P. C. (2019). *Software Testing: A Craftsman's Approach*. CRC Press.
3. Leung, H. (2020). *Software Testing Techniques*. World Scientific Publishing.
4. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
5. Gao, J., Chen, T. Y., & Chiu, D. K. W. (2018). "A Comparative Study of Conformance Testing Techniques." *IEEE Transactions on Software Engineering*, 44(6), 558-573.
6. Mylopoulos, J., Chung, L., & Yu, E. (2015). "From Object-Oriented to Goal-Oriented Requirements Analysis." *Communications of the ACM*, 58(1), 78-85.
7. Pressman, R. S. (2014). *Rekayasa Perangkat Lunak: Pendekatan Praktisi*. Penerbit Andi.
8. Sutojo, T. (2017). *Pengujian Perangkat Lunak*. Penerbit Informatika.
9. Sommerville, I. (2015). *Rekayasa Perangkat Lunak*. Penerbit Prenhallindo.

10. Nugroho, A. Y., & Rini, A. D. (2016). *Pengantar Pengujian Perangkat Lunak*. Penerbit Andi.
11. Wiryana, I. N., & Susanto, A. (2019). *Pengembangan Sistem Informasi: Teori dan Praktik*. Penerbit Andi.
12. Supriyanto, A. (2018). *Panduan Lengkap Pengujian Perangkat Lunak Berbasis ISO/IEC 29119*. Penerbit Andi.
13. Pessi, K. (2019). *Pemrograman Web: PHP, MySQL, dan JavaScript*. Penerbit Andi.
14. Sulistyo, S. (2020). *Implementasi Sistem Informasi: Panduan Praktis Menggunakan Metode OOAD*. Penerbit Informatika.
15. Oetomo, A. S. (2016). *Analisis dan Perancangan Sistem Informasi: Pendekatan Terstruktur Teori dan Praktik*. Penerbit Andi.

PENERBIT UNPRI PRESS
Jl. Sampul No.4, Sei Putih Barat,
Medan Petisah, Medan - 20118