TEKNIK MENJAMIN KUALITAS

BAGI PENGEMBANG PERANGKAT LUNAK

YAYASAN PRIMA AGUS TEKNIK

Sindhu Rakasiwi

TEKNIK MENJAMIN KUALITAS

Bagi Pengembang Perangkat Lunak

Sindhu Rakasiwi, S.Kom., M.Kom

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Teknik Menjamin Kualitas
Bagi Pengembang Perangkat Lunak

Penulis: Sindhu Rakasiwi, M.Kom

ISBN: **978-623-8120-19-2 (PDF)**

Editor: Myra Andriana, M.Si., M.Kom.

Penyunting: Tantik Sumarlin, S.Kom., M.Si

Desain Sampul dan Tata Letak: Yuli Fitrianto, S.T., M.Kom.

Penerbit:

Yayasan Prima Agus Teknik

Redaksi:

Jln Majapahit No 605 Semarang

Tlpn. (024) 6723456 Fax . 024-6710144

Email: penerbit_ypat@stekom.ac.id

Distributor Tunggal:

UNIVERSITAS STEKOM

Jalan Majapahit No. 605 Semarang

Tlpn. (024) 6723456 Fax. 024-6710144

Email: info@stekom.ac.id

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah SWT atas segala kemudahan yang telah diberikan, sehingga buku ini dapat diselesaikan. Buku Teknik Menjamin Kualitas Bagi Pengembang Perangkat Lunak ini merupakan buku panduan bagi mahasiswa untuk mempelajari mata kuliah penjamin mutu perangkat lunak.

Buku ajar teknik menjamin kualitas ini isinya mencakup seluruh uraian materi yang diperlukan bagi siapapun terutama bagi para mahasiswa dan pengembang perangkat lunak untuk menjamin kualitas atau mutu dari suatu perangkat lunak yang dikembangkan. Materi disajikan secara lengkap, mulai dari dasar pemahaman tentang penjaminan mutu, dimensi-dimensi dari kualitas, perangkat, veriikasi, validasi, tinjauan keandalan dan proses, hingga ke pandangan baru yang ada. Target penulisan buku ini tidak hanya sebagai referensi untuk kalangan mahasiswa, melainkan juga untuk umum. Buku ini dilengkapi dengan ringkasan materi serta soal-soal latihan pada bagian akhir di setiap bab.

Terima kasih diucapkan atas dukungan dan motivasi dari teman-teman dan keluarga. Segala masukkan dan kritik yang membangun diharapkan dari pembaca untuk perbaikan buku ini.

Semarang, Januari 2023

Sindhu Rakasiwi

DAFTAR ISI

BAB I
DASAR-DASAR PENJAMINAN MUTU

A. DEFINISI KUALITAS

Kita sering melihat kata kualitas digunakan sebagai istilah yang berdiri sendiri, tanpa ada kata sifat yang melekat padanya. Orang biasanya tidak menggunakan istilah kualitas yang baik untuk mengekspresikan kepuasan mereka dengan produk atau layanan yang mereka gunakan. Mengatakan bahwa produk tertentu adalah produk yang berkualitas menyiratkan bahwa produk tersebut berkualitas baik. Di sisi lain, orang pasti menggunakan istilah kualitas buruk untuk mengekspresikan ketidakpuasan mereka terhadap produk atau layanan yang mereka gunakan. Oleh karena itu, kata sifat baik secara implisit melekat pada kata kualitas di benak kebanyakan orang.

Untuk pelanggan atau pengguna akhir suatu produk, kualitas berkonotasi fungsi bebas cacat, keandalan, kemudahan penggunaan, tingkat toleransi kesalahan yang dapat diterima selama penggunaan, dan keamanan dari cedera pada orang atau properti. Untuk pelanggan atau pengguna akhir layanan, kualitas berkonotasi keandalan kinerja, kemudahan memperoleh layanan, layanan ahli, layanan yang menyenangkan, dan perlindungan dari kerusakan yang diakibatkannya. Untuk produsen barang, kualitas berkonotasi kesesuaian produk dengan spesifikasi, yang dapat ditentukan oleh badan pemerintah, asosiasi industri atau badan standar, atau oleh organisasi produsen itu sendiri.

Untuk penyedia layanan, kualitas berarti memenuhi tenggat waktu dan penyampaian layanan yang sesuai dengan spesifikasi dan standar pelanggan yang mungkin telah ditetapkan oleh badan pemerintah, asosiasi industri atau badan standar, atau oleh organisasi penyedia itu sendiri. Untuk badan pemerintah, kualitas berkonotasi keselamatan dan perlindungan konsumen dari penipuan. Untuk asosiasi industri atau badan standar, kualitas berkonotasi menjaga reputasi industri, melindungi industri dari penipuan dan tuntutan hukum, dan menangani masalah konsumen, badan pemerintah, dan industri itu sendiri.

Mengingat perbedaan di atas dalam arti kualitas, jelas bahwa kata tersebut memiliki banyak konotasi yang melekat padanya.

APA ITU KUALITAS?

Sebelum melangkah lebih jauh, pertama-tama kita perlu mendefinisikan kualitas kata dengan cara yang membahas semua konotasi yang disebutkan di atas. Organisasi Internasional untuk Standardisasi (ISO 9000, edisi kedua, 2000) mendefinisikan kualitas sebagai sejauh mana seperangkat karakteristik yang melekat memenuhi persyaratan. Kualitas dapat digunakan dengan kata sifat seperti miskin, baik, atau sangat baik. Inheren, sebagai lawan dari ditugaskan, berarti ada di dalam sesuatu, sebagai karakteristik permanen.

Definisi ini mengandung tiga istilah kunci: persyaratan, karakteristik, dan derajat. Persyaratan dapat dinyatakan oleh pelanggan dalam skenario yang dibuat berdasarkan pesanan atau oleh produk spesifikasi produk dalam skenario produk komersial yang siap pakai. Karakteristik mengacu pada kemampuan penyampaian atau, dengan kata lain, kekokohan (fitness) produk. Kata derajat menyiratkan bahwa kualitas adalah kontinum, dimulai dengan nol dan bergerak menuju, mungkin, tak terhingga. Kesimpulan ini, bagaimanapun, adalah ambigu dan mengarah pada persepsi yang salah. Apa tingkat di mana kualitas disebut "buruk" atau "baik" atau "sangat baik"? Lebih penting lagi, siapa yang berwenang mendefinisikan istilah “miskin”, “baik”, dan “sangat baik”?

Definisi kualitas lain yang populer, seperti yang didefinisikan oleh Joseph Moses Juran, adalah kesesuaian untuk digunakan, dengan kesesuaian dan penggunaan sangat penting untuk pemahaman yang tepat tentang kualitas. Kecuali kita mendefinisikan dua kata kunci ini, definisi kualitas sudah lengkap. Interpretasi konsumen dan interpretasi penyedia dari kedua istilah ini sering berselisih.

B. SPESIFIKASI

Karena kesesuaian dan penggunaan adalah istilah penting, mereka tidak dapat dibiarkan terbuka untuk interpretasi. Organisasi sering mendefinisikan kedua istilah ini dalam spesifikasi mereka untuk produk atau layanan yang mereka sediakan.

Mari kita lihat lebih dekat atribut spesifikasi yaitu spesifikasi eksplisit atau implisit. Eksplisit berarti penyedia memilih spesifikasi dan membuatnya tersedia bagi pelanggan. Tersirat berarti bahwa spesifikasi tidak didefinisikan tetapi dianggap perlu; contohnya termasuk persyaratan keselamatan, keamanan, dan toleransi kesalahan.Spesifikasi dapat ditentukan oleh penyedia atau badan eksternal, seperti organisasi pemerintah, asosiasi industri, atau badan standar. Mereka dibuat tersedia untuk pelanggan, dan mereka dipatuhi oleh penyedia.

Seringkali, penyedia menggunakan definisi spesifikasi yang tidak etis dan memberikan layanan atau produk yang dapat merugikan pelanggan dan mungkin juga industri. Hal ini mengakibatkan organisasi industri berkumpul untuk membentuk asosiasi, seperti asosiasi produsen dan penyedia layanan sebagai asosiasi, yang menentukan spesifikasi untuk produk atau layanan industri tertentu mereka. Pemerintah juga turun tangan dan membentuk badan standar, yang menetapkan spesifikasi untuk berbagai produk dan layanan. Departemen-departemen pertahanan di berbagai negara sering kali menetapkan spesifikasi untuk beragam produk yang akan digunakan oleh angkatan bersenjata mereka.

Spesifikasi ini menetapkan seperangkat standar minimum untuk dipatuhi oleh penyedia produk atau layanan, sehingga kesesuaian untuk digunakan ditentukan dan dipastikan.Spesifikasi yang ditetapkan secara formal tersebut menjadi standar industri dan disewakan oleh asosiasi industri kepada masyarakat umum dengan biaya nominal yang mencakup biaya produksi dan distribusi standar ini. Contoh badan yang mengeluarkan standar secara teratur termasuk Institut Standar Nasional Amerika, Institut Standar Inggris, Spesifikasi Layanan Gabungan, Deutsches Institut für Normung, ISO, Komisi Elektroteknik Internasional, Persatuan

Telekomunikasi Antar Nasional, Asosiasi Produsen Listrik Nasional, dan Institut dari Insinyur Listrik dan Elektronika. Sebagai pengakuan atas kontribusi mereka terhadap kualitas dan kesejahteraan konsumen secara umum, satu hari telah disisihkan setiap tahun untuk merayakan organisasi semacam itu: Hari Standar Dunia adalah 14 Oktober.

Komponen standar :

1. Atribut komponen yang membentuk suatu produk, yang dapat mencakup bahan yang digunakan dan dimensi serta metode pengujian produk

2. Tujuan penggunaan produk atau layanan

3. Batasan produk yang perlu disampaikan kepada pelanggan

4. Proses pembuatan komponen

5. Parameter keamanan dan keselamatan yang perlu dibangun

Memahami bahwa spesifikasi adalah inti dari kualitas, sekarang kita dapat mendefinisikan istilah tersebut dengan cara yang lebih meyakinkan. Selain itu, penting bahwa kualitas didefinisikan dari sudut pandang penyedia, karena penyedialah yang membangun kualitas menjadi produk atau layanan, dan di lokasi penyedia di mana kualitas dipastikan.

C. DEFINISI KUALITAS DARI SISI PENYEDIA

Kualitas adalah atribut produk atau layanan yang diberikan kepada konsumen yang sesuai atau melebihi spesifikasi terbaik yang tersedia untuk produk atau layanan tersebut. Ini termasuk membuat spesifikasi tersebut tersedia bagi pengguna akhir produk atau layanan.

Spesifikasi yang menjadi dasar produk atau layanan yang disediakan mungkin telah ditetapkan oleh badan pemerintah, industri sebagai:asosiasi, atau badan standar. Jika definisi tersebut tidak tersedia, penyedia dapat menentukan spesifikasinya.

Definisi kualitas ini mengamanatkan bahwa penyedia:

a. Tetapkan spesifikasi jika belum ditentukan oleh badan yang lebih tinggi, seperti badan pemerintah, asosiasi industri, atau organisasi standar

b. Patuhi definisi spesifikasi terbaik yang tersedia Pastikan kesesuaiannya 100% atau lebih baik—tidak kurang

c. Sediakan kepada pelanggan spesifikasi yang memastikan kesesuaiannya

Hasil dari suatu produk atau jasa yang memenuhi definisi kualitas di atas adalah pelanggan dapat menggunakan produk secara efektif selama hidupnya atau menikmati layanan

sepenuhnya. Hasil ini selanjutnya mengamanatkan bahwa penyedia bertanggung jawab untuk memberikan dukungan apa pun yang diperlukan oleh pelanggan untuk menikmati atau memanfaatkan produk atau layanan sepanjang hidupnya.

Setiap produk atau layanan yang memenuhi persyaratan definisi ini dinilai sebagai "produk/layanan berkualitas", dan produk atau layanan apa pun yang tidak memenuhi persyaratan definisi ini diberi peringkat "kualitas buruk".

D. KUALITAS DAN KEANDALAN

Kualitas dan keandalan saling terkait dan tidak dapat dipisahkan, tetapi apa yang dimaksud dengan reliabilitas ?Keandalan suatu produk adalah kemampuannya untuk berfungsi pada tingkat kinerja yang ditentukan selama masa pakainya.

Dua frasa penting dalam definisi ini:

1. Tingkat kinerja yang ditentukan—Tingkat kinerja ditentukan dalam spesifikasi produk atau layanan. Itu harus 100% atau lebih dari spesifikasi dan tidak kurang. Penggunaan terus menerus juga merupakan spesifikasi. Sebagai contoh, sebuah mobil mungkin mampu dikendarai dengan kecepatan 100 mil per jam, tetapi berapa lama sebuah mobil dapat bertahan dikendarai terus menerus pada kecepatan itu? Biasanya, kinerja didefinisikan pada dua tingkat: kinerja normal dan kinerja puncak.

2. Durasi masa pakainya—Durasi perlu ditentukan untuk kinerja normal serta kinerja puncak.

Sebuah produk memiliki dua kehidupan:

a. Kehidupan pertama atau kehidupan awal—Kehidupan awal, sebelum perbaikan menjadi diperlukan, biasanya ditentukan sebagai masa garansi atau garansi. Setelah berakhirnya masa pakai ini, perawatan rutin mungkin dilakukan diperlukan untuk mempertahankan kinerja pada tingkat yang ditentukan untuk produk.

b. Masa operasi —Jangka waktu setelah garansi berakhir,dengan asumsi pemeliharaan dilakukan. Setelah berakhirnya hidup ini,mungkin tidak ekonomis untuk mempertahankan produk agar beroperasi pada tingkat kinerja yang ditentukan.

Dengan kata lain, kualitas melibatkan penyampaian fungsionalitas yang ditentukan di bawah kondisi tertentu, dan keandalan melibatkan penyampaian fungsionalitas tertentu pada tingkat kinerja tertentu selama masa pakai produk, bahkan dengan sedikit penyimpangan dalam kondisi yang ditentukan.

Sementara masa pakai awal ditentukan oleh produsen sebagai masa garansi, kehidupan setelah masa garansi biasanya tidak ditentukan. Jika ya, itu ditentukan dengan ketentuan seperti "tergantung pada kondisi bahwa produk dipertahankan dan dilayani oleh teknisi ahli kami sendiri" atau yang serupa. Jika penyewa utama produk dipercayakan kepada pabrikan atau bengkel resminya, pabrikan menetapkan dua norma: waktu rata-rata antara kegagalan dan rata-rata waktu untuk memperbaiki.

Rata-rata waktu antara kegagalan adalah periode rata-rata antara dua berturut-turut kegagalan, dengan asumsi bahwa pemeliharaan yang tepat dilakukan setiap waktu dan pemeliharaan utama sesuai dengan ketentuan pabrikan. Hal ini dinyatakan dalam jumlah jam berjalan untuk produk. Rata-rata waktu untuk memperbaiki adalah waktu rata-rata diperlukan untuk mengembalikan produk ke fungsi aslinya dengan melakukan perbaikan yang diperlukan. Hal ini dinyatakan dalam jumlah jam yang dibutuhkan untuk perbaikanproduk. Keandalan diukur dengan dua ukuran ini.

Dalam hal perangkat lunak, pengamatan yang sering dilakukan adalah perangkat lunak tidak memiliki bagian yang bergerak yang menyebabkan produk rusak karena keausan. Satu kali produk perangkat lunak berfungsi pada tingkat kualitas dan fungsionalitas yang ditentukan, seharusnya tidak perlu ada pemeliharaan. Oleh karena itu, istilah keandalan harus tidak dapat diterapkan pada perangkat lunak. Namun, alasan ini benar hanya jika konfigurasi yang menjalankan produk perangkat lunak tetap tidak berubah. Jika konfigurasi perangkat keras dan perangkat lunak tidak berubah, tidak ada perbaikan yang diperlukan, sehingga atribut keandalan tidak dapat diterapkan. Hari-hari ini, bagaimanapun, banyak faktor lain yang berperan dalam kestabilan konfigurasi perangkat keras dan perangkat lunak. Berikut ini adalah beberapa situasi umum yang dapat mengubah konfigurasi perangkat keras dan perangkat lunak:

a. Sistem operasi baru memasuki pasar setiap tiga tahun.

b. Browser Web baru atau pembaruan untuk browser saat ini dirilis secara berkala.

c. Virus dan spyware baru dilepaskan di Internet tanpa curiga pengguna.

d. Komputer sering dibanjiri dengan sejumlah alat baru, mulai dari office suite hingga perangkat lunak antivirus hingga utilitas yang dapat diunduh.

e. Perubahan diperkenalkan ke tingkatan (middleware) di arsitektur multitier produk perangkat lunak masa depan.

f. Produk perangkat lunak dapat menggunakan pustaka bersama yang merupakan bagian dari perangkat lunak sistem yang disertakan bersama sistem operasi. Ada kemungkinan bahwa perpustakaan bersama ini diperbarui atau dimodifikasi.

g. Produk perangkat lunak dapat menggunakan pustaka kode pihak ketiga untuk melakukan fungsi khusus seperti pemrosesan aturan, kemandirian basis data, dll. Pustaka kode pihak ketiga ini dapat diperbarui atau dimodifikasi.

h. Menginstal dan menghapus instalan utilitas pada sistem dapat mengakibatkan perubahan atau penghapusan pustaka bersama yang digunakan oleh produk perangkat lunak.

Semua aktivitas ini mengubah konfigurasi sistem tempat produk perangkat lunak berjalan, dan di sinilah pertanyaan tentang keandalan perangkat lunak berperan. Sebuah produk perangkat lunak dikatakan dapat diandalkan jika dapat menahan patch kecil ke sistem operasi dan middleware.

Karena profesional kualitas perangkat lunak tidak dapat memprediksi pemutakhiran apa yang akan dilakukan pada perangkat lunak sistem (baik itu sistem operasi, database, browser, atau middleware), mereka tidak dapat menentukan keandalan perangkat lunak dalam jam kerja. Mereka juga mungkin tidak dapat menentukan waktu rata-rata antara kegagalan produk perangkat lunak dalam jam berjalan, karena produk perangkat lunak tidak gagal karena digunakan selama beberapa jam. Namun, itu bisa gagal karena perubahan dalam konfigurasi sistem. Demikian halnya dengan waktu rata-rata untuk perbaikan, karena perbaikan bukanlah untuk mengembalikan perangkat lunak ke kondisi sedekat mungkin, melainkan untuk menghilangkan dampak dari beberapa perubahan dalam konfigurasi sistem.

E. EVOLUSI KONSEP KUALITAS

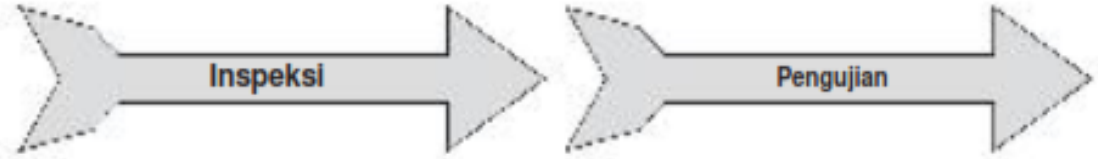
Meskipun kualitas adalah kata kuno, pemahamannya di tingkat organisasi telah berkembang belakangan ini, terutama sejak Perang Dunia II. Awalnya, dianggap bahwa hanya pengrajin yang dapat mencapai status produk "berkualitas". Namun, ketika Revolusi Industri memindahkan manufaktur dari toko pengrajin ke pabrik, dengan banyak pengrajin mengerjakan satu produk, supervisor menjadi sangat penting dalam mencapai produk yang berkualitas. Jika ada bagian yang hilang atau ada baut yang kendor, itu adalah kesalahan pengawas karena tidak menyadarinya. Ketika tekanan pada supervisor untuk memastikan kualitas meningkat, pengawasan yang sebenarnya mengambil kursi belakang, yang mempengaruhi produktivitas dan produksi.

Akhirnya manajemen sadar bahwa penunjukan inspektur independen diperlukan untuk memastikan bahwa setiap bagian dipasang dengan benar dan setiap baut dikencangkan. Maka muncullah profesi inspeksi, seiring dengan perkembangan sejumlah alat, teknik, dan metode inspeksi. Inspeksi menjadi area penelitian tersendiri. Contoh alat yang dikembangkan secara khusus untuk inspeksi yang sekarang menjadi standar di lingkungan manufaktur termasuk pengukur “go/no-go” dan jig inspeksi.

Inspeksi, sebagai mata rantai dalam rantai manufaktur (lihat Gambar 1.1), berfungsi dengan baik untuk beberapa waktu, tetapi menjadi tidak memadai karena fungsionalitas produk menjadi lebih bervariasi. Memastikan bahwa setiap bagian dipasang dengan benar dan setiap baut dikencangkan dengan benar segera ditemukan tidak memadai untuk memastikan berfungsinya produk dengan benar. Hal ini terutama berlaku untuk produk listrik seperti motor dan mesin, karena produk tersebut memerlukan pengujian fungsionalitas selain inspeksi keseluruhan. Disadari bahwa inspeksi saja tidak cukup untuk memastikan kualitas produk dan produk yang keluar dari pabrik juga harus diuji fungsinya.

Sekitar waktu yang sama, subkontrak pembuatan suku cadang ke pabrikan khusus mulai dilakukan, dimulai di industri otomotif. Ini membawa masalah baru: memastikan kualitas input. Dengan demikian, inspeksi ke dalam (inspeksi suku cadang yang diterima dari pemasok dan subkontraktor) dan pengujian juga muncul. Manufaktur batch dan pekerjaan juga mulai muncul pada waktu yang bersamaan, menghasilkan konsep baru: kontrol kualitas (lihat Gambar 1.2). Sejumlah cahaya baru era pada metode pengendalian kualitas muncul, termasuk pengambilan sampel inspeksi, kontrol kualitas statistik, grafik kontrol, dan sebagainya.

Gambar 1.1 Inspeksi



Gambar 1.2 Kontrol kualitas

Hingga saat ini, penekanan dalam hal kualitas adalah memastikan kualitas manufaktur. Ketika persaingan meningkat di antara produsen dan ketika atau organisasi mulai menyediakan produk serupa dan mungkin lebih baik menemukan bahwa produk dapat gagal karena cacat desain, bahkan jika kualitas produksi pabrik dikontrol dengan ketat. Salah satu contoh yang terlintas dalam pikiran adalah dua merek skuter roda dua: Vespa dan Lambretta. Keduanya mirip dalam hal tenaga kuda dan dalam spesifikasi mampu menampung dua orang, namun mereka berbeda dalam desain. Vespa menggunakan transmisi daya berbasis poros, sedangkan Lambretta menggunakan transmisi daya berbasis rantai. Lambretta akhirnya ditutup, sementara Vespa masih beroperasi sampai sekarang. Kurangnya popularitas skuter Lambretta bukan karena masalah kualitas manufaktur. Sebaliknya, itu adalah masalah desain, dan karena itu masalah yang sangat mempengaruhi kelangsungan hidup organisasi itu sendiri. Kualitas desain produk, yang bergantung pada spesifikasi yang ditetapkan untuk produk itu, sama pentingnya, jika tidak lebih, penting. untuk keberhasilan produk seperti halnya kontrol kualitas selama tahap pembuatan pabrik.

Untuk mencapai desain yang lebih baik, produsen perlu menetapkan pedoman desain, memanfaatkan pengalaman dan pengetahuan tentang organisasi dalam industri tertentu, serta umpan balik dari lapangan (keluhan pelanggan, pengamatan personel pemeliharaan, dan mempelajari produk pesaing). Hal ini mengakibatkan pengembangan standar dan pedoman untuk memastikan kualitas desain dan spesifikasi. Tinjauan desain diikuti untuk memastikan kualitas dalam desain produk.

Sementara aspek desain produk ini pasti termasuk dalam arena kualitas, itu di luar kapasitas departemen kendali mutu organisasi. Ini pengembangan memunculkan konsep jaminan kualitas (seperti yang digambarkan pada Gambar 1.3), merupakan bagian integral dari manufaktur yang mencakup inspeksi, pengujian, dan standar untuk desain.

Ada kesalahpahaman dalam industri pengembangan perangkat lunak bahwa kualitas jaminan berarti pengujian. Saya tidak yakin bagaimana kesalahpahaman ini terjadi. Pengujian

adalah pengujian; penjaminan mutu meliputi pemeriksaan (verifikasi), pengujian, dan standar. Dalam daftar istilah Integrasi Model Kematangan Kemampuan model dokumen untuk pengembangan (versi 1.2, 2006), jaminan kualitas didefinisikan sebagai “cara yang terencana dan sistematis untuk memastikan manajemen bahwa standar, praktik, prosedur, dan metode proses yang ditetapkan adalah terapan."

Satu kejadian catatan yang memiliki dampak signifikan pada evolusi konsep kualitas adalah transformasi yang dialami organisasi manufaktur Jepang. Pabrikan Jepang bangkit dari reputasi mereka sebagai pemasok barang murah berkualitas buruk untuk menjadi pemasok produk berkualitas tinggi. Dia adalah transformasi fenomenal, dan studi yang dilakukan pada metode fabrikasi pabrik Jepang dipublikasikan secara luas. Beberapa metode ini termasuk kualitas lingkaran kontrol, nol cacat, dan benar pertama kali.

Salah satu teknik Jepang yang diadopsi secara luas oleh produsen di seluruh dunia dunia adalah lingkaran kontrol kualitas, atau hanya lingkaran kualitas, sebagaimana mereka dikenal secara populer. Lingkaran kualitas adalah asosiasi sukarela pekerja dari yang sama fasilitas yang bertemu untuk membahas masalah terkait kualitas di fasilitas mereka dan muncul dengan kemungkinan solusi untuk meningkatkan kualitas. Jika diskusi mereka menunjukkan cacat, bersama-sama mereka menemukan solusi, mencobanya sebagai percontohan dan mempresentasikan hasilnya kepada manajemen. Jika manajemen puas dengan proposal tersebut, itu diimplementasikan, dan anggota lingkaran kualitas yang muncul dengan saran dihargai.

Dilaporkan bahwa industri manufaktur Jepang sangat diuntungkan dari lingkaran kualitas ini, dan manfaat ini dirasakan di seluruh dunia dalam berupa barang-barang yang ditingkatkan dari Jepang, negara yang sekarang dikenal dengan kualitasnya yang tinggi produk. Sementara konsep lingkaran kualitas disambut baik oleh ekonomi lain, implementasinya tidak menghasilkan hasil yang spektakuler di tempat lain.India khususnya dengan sepenuh hati mengadopsi konsep ini dan menghabiskan cukup banyak sumber daya untuk mengimplementasikannya, tetapi tidak mencapai hasil yang nyata atau nyata. hasil yang dapat diaudit, positif, dan sepadan.

Namun, apa yang dilakukan oleh transformasi kualitas produk Jepang hasilnya adalah kesadaran bahwa kualitas bukan hanya tanggung jawab kualitas departemen saja; itu adalah masalah organisasi. Jika kualitas diabaikan, kelangsungan hidup organisasi mungkin dipertaruhkan. Kesadaran ini mengarah pada pengembangan konsep manajemen kualitas total, yang mengharuskan seluruh organisasi terlibat dalam pencapaian kualitas-tidak hanya dalam hal hasil, tetapi dalam setiap aktivitas organisasi. Organisasi dipandang sebagai budaya-budaya yang didasarkan pada kualitas yang memandang kualitas sebagai unsur penting dalam semua aktivitasnya.

Perkembangan teknologi menciptakan dimensi baru untuk membantu mencapai kualitas: robot. Jepang kembali memimpin dan menggunakan robot secara ekstensif di pabrik-pabriknya. Karena kemungkinan kesalahan manusia telah dihapus dari sejumlah besar operasi, kemungkinan cacat dihilangkan. Dengan demikian, kebutuhan akan inspeksi menjadi marginal, meskipun pengujian tetap penting. Itu tidak mungkin untuk memeriksa semuanya. Ambil, misalnya, perakitan gearbox. Setelah dirakit, bagian dalam tidak dapat diperiksa kecuali

gearbox dibuka, tetapi jika dibuka, itu harus dipasang kembali. Hal ini memunculkan konsep kualitas proses, yang menanamkan konsep kualitas ke dalam proses manufaktur itu sendiri.

F. MANAJEMEN KUALITAS TOTAL

Konsep kualitas yang paling populer dalam industri manufaktur saat ini adalah total quality management (TQM). Hampir semua perusahaan manufaktur yang dikelola secara profesional telah menerapkan TQM dan mempraktekkannya dengan tekun. Industri pengembangan perangkat lunak, sadar atau tidak, melompat ke TQM melalui sertifikasi kualitas proses seperti ISO dan CMM. ISO mendefinisikan TQM sebagai "pendekatan manajemen untuk sebuah organisasi, berpusat pada kualitas, berdasarkan partisipasi semua anggotanya, dan bertujuan untuk kesuksesan jangka panjang melalui kepuasan pelanggan dan manfaat bagi semua anggota organisasi dan masyarakat."kualitas ke dalam proses manufaktur itu sendiri.

TQM adalah inisiatif kualitas di seluruh organisasi, yang berarti melibatkan seluruh organisasi dalam pengelolaan kualitas. Salah satu tujuan utama TQM adalah untuk mengurangi variasi proses dalam organisasi. Di Jepang, TQM mencakup empat langkah:

1. Kaizen—Fokus pada perbaikan proses berkelanjutan, untuk membuat setiap proses dalam organisasi terlihat, dapat diulang, dan terukur.

2. Atarimae hinshitsu—Keyakinan bahwa produk akan berfungsi sebagaimana dirancang untuk berfungsi.

3. Kansei—Mempelajari cara pengguna menggunakan produk, untuk memfasilitasi peningkatan produk.

4. Miryoketuki hinshitsu—Keyakinan bahwa produk harus memiliki nilai estetika dan kegunaan. Misalnya, mobil perlu terlihat menarik selain kemampuannya untuk mengangkut orang.

TQM menganjurkan standar kualitas dalam semua aspek fungsi organisasi, sertafilosofi "lakukan dengan benar pertama kali." Ini juga merekomendasikan penghapusan limbah dalam segala bentuknya. Seperti yang ada saat ini, TQM diadopsi sampai tingkat tertentu dalam organisasi yang memiliki jaminan kualitas di hati mereka, dengan inspeksi, pengujian, dan standar yang diterapkan secara menyeluruh dan konsisten.Meskipun konsep kualitas dikembangkan dalam organisasi manufaktur, semua konsep yang dibahas di atas juga relevan dengan organisasi pengembangan perangkat lunak.

G. RINGKASAN

Kualitas adalah atribut produk atau layanan yang diberikan kepada konsumen yang sesuai atau melebihi spesifikasi terbaik yang tersedia untuk produk atau layanan tersebut. Ini termasuk membuat spesifikasi tersebut tersedia bagi pengguna akhir produk atau layanan. TQM adalah inisiatif kualitas di seluruh organisasi, yang berarti melibatkan seluruh organisasi dalam pengelolaan kualitas. TQM menganjurkan standar kualitas dalam semua aspek fungsi organisasi, sertafilosofi "lakukan dengan benar pertama kali." Ini juga

merekomendasikan penghapusan limbah dalam segala bentuknya. Seperti yang ada saat ini, TQM diadopsi sampai tingkat tertentu dalam organisasi yang memiliki jaminan kualitas di hati mereka, dengan inspeksi, pengujian, dan standar yang diterapkan secara menyeluruh dan konsisten.Meskipun konsep kualitas dikembangkan dalam organisasi manufaktur, semua konsep yang dibahas di atas juga relevan dengan organisasi pengembangan perangkat lunak

H. SOAL LATIHAN

1) Apa yang dimaksud dengan kualitas?
2) Sebutkan empat cakupan TQM!
3) Sebutkan 5 komponen standard kualitas!
4) Apa saja yang dapat mengubah konfigurasi perangkat keras dan perangkat lunak?Sebutkan dan jelaskan!
5) Apa yang dimaksud dengan reliabilitas?

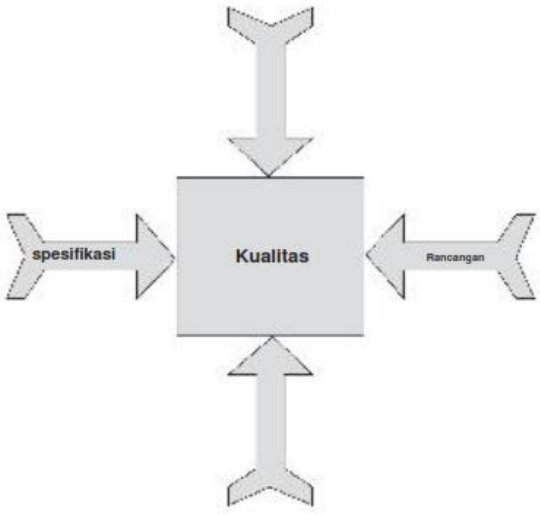
BAB II
EMPAT DIMENSI KUALITAS

A. EMPAT DIMENSI KUALITAS

Kualitas memiliki empat dimensi (seperti yang digambarkan pada Gambar 2.1):

1. Kualitas spesifikasi
2. Kualitas desain
3. Kualitas pengembangan (konstruksi perangkat lunak)
4. Kualitas kesesuaian

Spesifikasi adalah titik awal dalam perjalanan penyediaan produk atau layanan, diikuti dengan desain dan kemudian pengembangan. Kualitas kesesuaian memastikan seberapa baik kualitas itu dibangun ke dalam penyampaian di setiap tahap. Dimensi ini dibahas secara rinci dalam bab ini.



Gambar 2.1 Empat Dimensi Kualitas

B. KUALITAS SPESIFIKASI

Spesifikasi kualitas mengacu pada seberapa baik spesifikasi didefinisikan untuk produk atau layanan yang disediakan. Spesifikasi tidak memiliki aktivitas pendahulunya, dan semua aktivitas lainnya berhasil memenuhi spesifikasi. Jadi,

jika spesifikasinya lemah, desainnya akan lemah, yang mengakibatkan pengembangan dan pembuatan produk yang lebih rendah atau salah, dan upaya yang dihabiskan untuk memastikan bahwa kualitas terpasang akan sia-sia. Spesifikasi biasanya harus mencakup enam aspek berikut:

a. Aspek fungsionalitas —Tentukan fungsi apa yang akan dicapai oleh produk atau layanan.
b. Aspek kapasitas —Tentukan muatan yang dapat dibawa produk (seperti 250 penumpang di pesawat atau 100 pengguna bersamaan untuk aplikasi Web) atau jumlah orang yang dapat dilayani oleh layanan.
c. Aspek penggunaan yang dimaksudkan —Tentukan kebutuhan atau kebutuhan produk atau layanan wakil memuaskan.
d. Aspek reliabilitas —Menentukan berapa lama produk dapat dinikmati sebelum memerlukan pemeliharaan, atau kepastian penyampaian layanan dan kesesuaian dengan persyaratan pengguna.
e. Aspek keselamatan —Tentukan tingkat ambang batas untuk memastikan keselamatan orang dan properti dari penggunaan produk atau layanan.
f. Aspek keamanan —Tentukan ancaman apa pun yang perlu disiapkan oleh produk atau layanan.

Bagaimana kami memastikan bahwa kami memiliki spesifikasi yang lengkap dan benar? Aspek pertama untuk memastikan bahwa spesifikasi dibuat dengan benar adalah dengan melibatkan orang-orang yang memenuhi syarat, seperti analis bisnis atau analis sistem, untuk melaksanakan pekerjaan tersebut. Para profesional ini harus dilatih dengan baik untuk melaksanakan rekayasa persyaratan. Aspek kedua adalah mengembangkan standar internal atau mengadopsi standar asosiasi profesional atau badan standar yang harus diikuti oleh para analis. Standar-standar ini menetapkan tingkat minimum dalam menyusun spesifikasi.

Awalnya, spesifikasi harus dikembangkan untuk produk dengan cara biasa. Dalam skenario proyek yang digerakkan oleh pelanggan internal atau eksternal, persyaratan pengguna dikumpulkan. Dalam skenario produk komersial, persyaratan dikumpulkan dari latihan survei pasar.

Setelah persyaratan telah dikumpulkan, mereka perlu dikembangkan. Ini melibatkan pemisahan persyaratan menjadi persyaratan fungsional, persyaratan kegunaan, persyaratan keselamatan dan keamanan, persyaratan keandalan, dan sebagainya. Persyaratan ini juga harus diperiksa terhadap standar organisasi untuk kegunaan, keselamatan, dan keamanan, dan persyaratan yang hilang harus diisi. Kemudian, setiap kelas persyaratan dianalisis untuk kelengkapan baik dengan latar belakang produk yang ada atau proyek masa lalu. Jika keduanya tidak tersedia, ahli fungsional harus meneliti dan mengisi persyaratan yang hilang.

Jika akses ke ahli tidak tersedia, maka tim di dalam organisasi dibentuk untuk melakukan latihan brainstorming untuk memastikan bahwa spesifikasinya komprehensif di semua kelas. Dalam skenario produk komersial, survei pasar kedua untuk memanfaatkan calon pengguna dapat dilakukan untuk meningkatkan spesifikasi.

C. KUALITAS DESAIN

Kualitas desain mengacu pada seberapa baik produk atau layanan yang akan dikirimkan dirancang. Tujuan untuk desain adalah untuk memenuhi spesifikasi yang ditetapkan untuk produk atau jasa yang disediakan. Desain menentukan bentuk dan kekuatan produk atau jasa. Oleh karena itu, jika desainnya lemah, produk atau jasa akan gagal, bahkan jika spesifikasinya sudah ditentukan dengan sangat baik. Meskipun desain adalah aktivitas kreatif, dapat dibagi menjadi dua fase: desain konseptual dan rekayasa. Desain konseptual memilih pendekatan solusi dari berbagai pendekatan yang tersedia. Teknik menggunakan pendekatan yang dipilih dan mengerjakan detail untuk mewujudkan solusi. Desain konseptual adalah bagian kreatif dari proses, dan rekayasa adalah bagian detail.

Mari kita gunakan desain jembatan sebagai contoh untuk mengilustrasikan perbedaan antara desain konseptual dan teknik. Jembatan dapat berupa jembatan sederhana atau jembatan gantung. Sebuah jembatan sederhana memiliki sejumlah pilar (kolom) dengan jarak yang sama yang menopang jembatan dan lalu lintas yang mengalir di atasnya. Sebuah jembatan gantung memiliki pilar di setiap ujungnya, dengan kabel yang ditarik dari kedua pilar ini untuk menopang jembatan. Untuk kelas jembatan ini, ada banyak alternatif untuk material suspensi, lokasi pilar, desain pilar, desain kabel suspensi, dan sebagainya. Desain konseptual memutuskan aspek-aspek ini. Desain teknik mengerjakan detail seperti dimensi untuk setiap komponen, pemilihan bahan, metode penyambungan, dan sebagainya.

Dalam hal perangkat lunak, desain konseptual mengacu pada arsitektur perangkat lunak, navigasi, jumlah tingkatan, pendekatan terhadap fleksibilitas, portabilitas, pemeliharaan, dan sebagainya. Desain teknik mengacu pada desain database, spesifikasi program, desain layar, desain laporan, dll. Desain perangkat lunak biasanya berisi elemen-elemen berikut:

1. Desain fungsionalitas
2. Arsitektur perangkat lunak
3. Navigasi
4. Desain basis data
5. Platform pengembangan
6. Platform penerapan
7. Desain antarmuka pengguna
8. Desain laporan
9. Keamanan
10. Toleransi kesalahan
11. Kapasitas
12. Keandalan

13. Pemeliharaan

14. Efisiensi dan kesesuaian

15. Kopling dan kohesi

16. Spesifikasi program

17. Desain pengujian

Bagaimana kita memastikan bahwa desain yang tepat dipilih dan diimplementasikan? Seperti halnya kualitas spesifikasi, sebelum desain perangkat lunak dicoba, adalah penting bahwa orang-orang yang memenuhi syarat, terlatih dalam seni dan ilmu desain perangkat lunak, sudah siap.Standar desain perangkat lunak dikembangkan sendiri atau, sebagai alternatif, diadopsi dari asosiasi profesional atau badan standar. Standar ini membantu desainer untuk mencapai desain terbaik.

Adalah normal untuk melakukan sesi brainstorming di awal proyek desain perangkat lunak, untuk memilih satu alternatif desain yang optimal dan untuk memutuskan aspek desain secara keseluruhan, seperti jumlah tingkatan, platform teknologi, kopling dan kohesi perangkat lunak, dan lain-lain. Sesi brainstorming membantu desainer sampai pada solusi terbaik untuk proyek yang ada. Prototipe desain dapat dibuat dan dievaluasi, yang merupakan praktik normal khususnya dalam kasus pengembangan produk komersial yang siap pakai.

Desain akhir kemudian dievaluasi terhadap standar organisasi untuk memastikan bahwa desain akan bekerja untuk proyek tersebut. Desain harus ditinjau dari rekan sejawat, pakar, dan manajer seperti yang dipersyaratkan sebelum melaksanakan desain detail seluruh produk.

D. KUALITAS PENGEMBANGAN (KONSTRUKSI PERANGKAT LUNAK)

Dalam bidang tertentu, tidak mungkin kualitas dapat diuji tanpa merusak produk itu sendiri. Misalnya, ketebalan dan kelengketan cat pada permukaan tidak dapat dipastikan tanpa merusak cat itu sendiri. Berbagai poros yang digunakan dalam mobil ditempa dan diberi perlakuan panas untuk membuatnya lebih kuat, dan praktis tidak ada cara untuk mengujinya untuk memastikan bahwa kualitas yang diinginkan terpasang tanpa merusak poros. Dalam kasus seperti itu, inspeksi dalam proses dilakukan untuk memastikan bahwa proses dipatuhi dengan rajin dan beberapa sampel dikenai pengujian destruktif.

Untungnya, ketika menyangkut perangkat lunak, tidak ada yang perlu dihancurkan selama pengujian, dan koreksi dapat dilakukan tanpa kerugian materi, tetapi pengujian membutuhkan waktu lebih lama untuk dilakukan di bidang pengembangan perangkat lunak daripada di manufaktur. Inspeksi dan pengujian hanya membutuhkan sebagian kecil dari waktu yang diperlukan untuk membuat bagian atau produk dalam manufaktur, tetapi pengujian perangkat lunak terkadang membutuhkan lebih banyak waktu dan upaya daripada yang diperlukan untuk mengembangkan perangkat lunak.

Biasanya, aktivitas berikut merupakan bagian dari pengembangan perangkat lunak:

1. Buat database dan struktur tabel
2. Mengembangkan perpustakaan yang terhubung secara dinamis untuk rutinitas umum
3. Mengembangkan layar
4. Mengembangkan laporan
5. Mengembangkan rencana pengujian unit
6. Mengembangkan rutinitas proses terkait untuk semua aspek lain, seperti keamanan, efisiensi, toleransi kesalahan, dan lain-lain

Konstruksi berkualitas baik dicapai dengan mengikuti pedoman pengkodean bahasa pemrograman yang digunakan. Biasanya ada pedoman pengkodean terpisah untuk setiap bahasa pemrograman yang digunakan dalam suatu organisasi. Merupakan kebiasaan untuk menentukan pedoman pengkodean sebelum mulai menulis program dalam suatu bahasa. Pedoman pengkodean berisi konvensi penamaan, pemformatan kode, pedoman efisiensi, dan pedoman pencegahan cacat yang membantu pengembang menulis kode yang andal dan bebas cacat. Tentu saja, sangat penting untuk memiliki orang-orang yang memenuhi syarat yang terlatih dalam pengembangan perangkat lunak. Konstruksi mengikuti desain perangkat lunak, dan harus selalu sesuai dengan dokumen desain. Dengan cara ini, kualitas yang baik dalam konstruksi dapat dicapai.

E. KUALITAS KESESUAIAN

Kualitas kesesuaian berkaitan dengan seberapa baik suatu organisasi memastikan bahwa kualitas dibangun ke dalam produk melalui tiga dimensi di atas. Ini adalah satu hal untuk melakukan pekerjaan yang berkualitas, tetapi itu adalah hal lain untuk menemukan cacat yang bersembunyi di produk kerja dan memastikan bahwa produk berkualitas baik memang dibuat. Pada dasarnya, kualitas kesesuaian memeriksa seberapa baik pengendalian kualitas dilakukan dalam organisasi.

Bagaimana kita menentukan seberapa baik suatu organisasi melakukan kegiatan yang memastikan bahwa kualitas memang dibangun menjadi hasil? Salah satu cara untuk memastikan kemanjuran kegiatan jaminan kualitas adalah dengan menggunakan seperangkat metrik kualitas. Metrik ini mencakup efisiensi penghilangan cacat dari setiap aktivitas kontrol kualitas, kualitas produk, dan kepadatan cacat. Cara lain untuk memastikan kemanjuran kegiatan jaminan kualitas adalah untuk membandingkan data benchmark industri untuk metrik kualitas dengan metrik organisasi. Lampiran G mencakup metrik dan pengukuran kualitas secara lebih rinci.

Buku ini membahas bagaimana membangun kualitas menjadi deliverable dan memastikan bahwa kualitas dibangun ke dalam tiga dimensi pertama yang disebutkan (spesifikasi perangkat lunak, desain, dan konstruksi), serta memastikan kualitas kesesuaian itu sendiri melalui pengukuran dan metrik kualitas.

F. MEMASTIKAN KUALITAS DALAM SPESIFIKASI

Ini adalah aktivitas pertama dalam membangun produk atau layanan. Tak perlu dikatakan, itu adalah kegiatan kreatif. Dalam industri perangkat lunak, spesifikasi disebut sebagai kebutuhan pengguna. Artinya, pengguna akhir produk perangkat lunak menganggap mereka sebagai persyaratan untuk produk yang diusulkan. Berikut ini adalah skenario yang mungkin untuk mendapatkan persyaratan pengguna:

1. Seorang analis bisnis melakukan studi kelayakan, menulis laporan, dan menyusun kebutuhan pengguna. Analis :

a. Bertemu dengan semua pengguna akhir dan mencatat kebutuhan mereka dan kekhawatiran

b. Bertemu dengan kepala fungsi dan mencatat kebutuhan mereka dan kekhawatiran

c. Bertemu dengan personel manajemen dan mencatat persyaratan dan kekhawatiran mereka

d. Mengkonsolidasikan persyaratan dan menyajikannya untuk memilih pengguna akhir, kepala fungsi, dan personel manajemen dan menerima umpan balik mereka. Menerapkan umpan balik dan menyelesaikan spesifikasi

2. Seperangkat persyaratan pengguna yang siap disajikan sebagai bagian dari permintaan untuk usul

3. Permintaan proposal menunjuk ke produk serupa dan meminta replikasi dengan penyesuaian khusus klien

Terlepas dari skenarionya, setelah spesifikasi siap, jaminan kualitas masuk. Peran jaminan kualitas di area ini adalah untuk memastikan bahwa spesifikasi lengkap dan mencakup semua area, termasuk fungsionalitas, kapasitas, keandalan, keselamatan, keamanan, tujuan penggunaan, dll.

Alat-alat untuk membangun kualitas ke dalam spesifikasi adalah sebagai berikut:

1. Dokumentasi proses—Rincian metodologi untuk mengumpulkan, mengembangkan, menganalisis, dan menyelesaikan spesifikasi

2. Standar dan pedoman, format, dan templat—Menentukan set minimum spesifikasi yang perlu dibuat dalam

3. Daftar Periksa—Membantu analis untuk memastikan kelengkapan spesifikasi fiksi

Dengan menggunakan alat ini, analis dapat mengembangkan spesifikasi yang komprehensif dan jelas untuk melakukan aktivitas berikutnya (yaitu desain) dan yang memastikan kualitas dibangun ke dalam spesifikasi.

G. MEMASTIKAN KUALITAS DALAM DESAIN

Sederhananya, proses desain adalah mengubah spesifikasi produk (atau persyaratan pengguna) menjadi dokumen desain yang dapat digunakan oleh programmer untuk mengembangkan kode sumber yang diperlukan untuk produk yang sedang dibangun. Biasanya, desain perangkat lunak adalah proses dua langkah:

1. Desain konseptual —Disebut sebagai desain tingkat tinggi, spesifikasi desain fungsional, spesifikasi kebutuhan perangkat lunak, dan desain arsitektur perangkat lunak. Pada langkah ini, arsitektur keseluruhan produk perangkat lunak, termasuk jumlah tingkatan, modul, pendekatan untuk mencapai fungsionalitas, desain basis data, ketahanan, keandalan, dan keamanan, ditentukan dan didokumentasikan. Dokumen ini digunakan oleh para desainer untuk melaksanakan desain rekayasa.

2. Desain rekayasa —Disebut sebagai desain tingkat rendah, spesifikasi desain detail, deskripsi desain perangkat lunak, dan desain program perangkat lunak. Pada langkah ini, spesifikasi rinci dibuat untuk setiap unit program, layar, laporan, tabel, dll., dan pemrogram menggunakan dokumen ini untuk mengembangkan kode sumber.

Alat untuk membangun kualitas ke dalam desain meliputi:

1. Dokumentasi proses—Rincian metodologi untuk desain alter native yang harus dipertimbangkan, kriteria untuk memilih alternatif untuk proyek, dan finalisasi desain konseptual.
2. Standar dan pedoman, format, dan templat—Tentukan arsitektur perangkat lunak yang memungkinkan beserta kelebihan dan kekurangannya, metodologi untuk daftar pendek alternatif desain, dan sebagainya.
3. Daftar Periksa—Bantu desainer untuk memastikan bahwa desain dilakukan secara komprehensif dan tepat.

Dengan menggunakan alat ini, desainer dapat mengembangkan desain yang komprehensif, jelas untuk melakukan aktivitas selanjutnya (yaitu konstruksi perangkat lunak), dan memastikan bahwa kualitas dibangun ke dalam desain.

H. MEMASTIKAN KUALITAS DALAM PEMBANGUNAN (KONSTRUKSI PERANGKAT LUNAK)

Pengembangan adalah tindakan membangun produk perangkat lunak sesuai dengan desain. Pada tahap ini, kode sumber dikembangkan dan dihubungkan dengan pustaka kode yang sudah ada sebelumnya, untuk melengkapi kode yang diperlukan untuk produk. Kode ini diubah menjadi kode yang dapat dieksekusi yang dapat dijalankan pada perangkat keras yang

dipilih. Ini juga merupakan tahap di mana database dirancang dan dibangun, sehingga data dapat dimuat dan digunakan oleh perangkat lunak.

Bagaimana kualitas dibangun ke dalam produk selama tahap pengembangan? Itu dibangun dengan mengikuti standar organisasi untuk kualitas kode serta pedoman pengkodean untuk bahasa pengembangan yang digunakan. Perubahan yang tidak terkontrol dapat merusak kualitas kode. Oleh karena itu, manajemen perubahan dan manajemen konfigurasi dianggap penting untuk memastikan kualitas kode.

I. MEMASTIKAN KUALITAS KESESUAIAN

Memastikan bahwa kualitas kesesuaian berada pada tingkat yang diinginkan dalam organisasi dicapai melalui pengukuran dan metrik kualitas. Efisiensi penghilangan cacat dari kegiatan verifikasi dan validasi, laju injeksi cacat, dan kepadatan cacat semuanya digunakan untuk tujuan ini. Selain itu, proyek-proyek dijadikan acuan pada tingkat organisasi dan analisis tren dilakukan. Metode ini terperinci dalam Lampiran G. Audit juga dilakukan untuk memastikan bahwa proyek sesuai dengan berbagai standar yang berlaku untuk kualitas bangunan ke dalam semua kegiatan, termasuk spesifikasi dan desain. Selain itu, data organisasi dibandingkan dengan tolok ukur industri, dan tindakan korektif atau pencegahan diambil untuk memastikan bahwa kesesuaian organisasi memang setara dengan industri.

Kualitas kesesuaian dibangun melalui definisi proses dan peningkatan berkelanjutan untuk semua aktivitas pengembangan perangkat lunak serta jaminan kualitas. Kualitas kesesuaian dipastikan melalui metrik dan pengukuran.

Tabel 2.1 merangkum teknik yang tersedia untuk memastikan kualitas di setiap dari empat dimensi kualitas.

Tabel 2.1. Teknik kesesuaian untuk empat dimensi kualitas

Dimensi Kualitas	Bagaimana Membangun Kualitas	Teknik Untuk Memastikan Kualitas
Kualitas spesifikasi	Dokumentasi proses pengembangan spesifikasi; standard dan pedoman, format, dan pelat suhu untuk menentukan spesifikasi; dan daftar periksa	Ulasan ahli, ulasan sejawat, dan curah pendapat
Kualitas desain	Dokumentasi proses desain perangkat lunak; standar dan garis panduan, format, dan template untuk desain perangkat lunak; dan	Tinjauan ahli, tinjauan sejawat, tinjauan manajerial, dan curah pendapat

	daftar periksa	
Kualitas pembangunan	Pedoman pengkodean, manajemen konfigurasi, dan manajemen perubahan	Ulasan sejawat dan perangkat lunak pengujian
Kesesuaian kualitas	Penerapan yang sungguh-sungguh dari semua aktivitas penjaminan mutu dalam organisasi, definisi proses, dan peningkatan	Audit, pengukuran, dan metrik untuk aktivitas jaminan kualitas, dan pembandingan metrik organisasi dengan metrik industri

J. RINGKASAN

Terdapat empat dimensi di dalam kualitasKualitas memiliki empat dimensi yaitu : kualitas spesifikasi , kualitas desain , kualitas pengembangan (konstruksi perangkat lunak) ,dan kualitas kesesuaian. Spesifikasi kualitas mengacu pada seberapa baik spesifikasi didefinisikan untuk produk atau layanan yang disediakan. Kualitas desain mengacu pada seberapa baik produk atau layanan yang akan dikirimkan dirancang. Kualitas pengembangan perangkat lunak ada beberapa aktivitas yaitu buat database dan struktur table,mengembangkan perpustakaan yang terhubung secara dinamis untuk rutinitas umum , mengembangkan layar , mengembangkan laporan, mengembangkan rencana pengujian unit, mengembangkan rutinitas proses terkait untuk semua aspek lain, seperti keamanan, efisiensi, toleransi kesalahan, dan lain-lain. Kualitas kesesuaian berkaitan dengan seberapa baik suatu organisasi memastikan bahwa kualitas dibangun ke dalam produk melalui tiga dimensi di atas.

K. SOAL LATIHAN

1) Sebutkan 4 dimensi kualitas!
2) Sebutkan 17 elemen desain perangkat lunak!
3) Jelaskan mengenai teknik kesesuaian untuk empat dimensi kualitas!
4) Jelaskan perbedaan antara desan konseptual dan desain rekayasa!
5) Sebutkan alat untuk membangun kualitas ke dalam desain !

BAB III
PERANGKAT LUNAK KUALITAS PRODUK

A. STANDPOINT FUNGSIONALITAS

Kualitas produk perangkat lunak memiliki banyak atribut. Dalam bab ini, kita akan memeriksa atribut-atribut ini, dimulai dengan yang paling penting. Atribut dasar dan terpenting dari produk perangkat lunak atau produk lainnya, dalam hal ini adalah bahwa ia memberikan fungsionalitas yang ditentukan secara akurat. Sebuah produk memiliki dua jenis fungsionalitas: fungsionalitas inti dan fungsionalitas tambahan.

B. FUNGSIONALITAS INTI

Fungsionalitas inti mengacu pada fungsionalitas utama yang dirancang untuk dipenuhi oleh produk, yang tanpanya produk tidak berguna. Misalnya, fungsi inti mobil adalah untuk mengangkut orang dari satu tempat ke tempat lain. Jika mobil tidak bisa melakukan itu,itu tidak berguna, tidak peduli betapa menarik atau nyamannya itu.

Dalam hal produk perangkat lunak, pertimbangkan, misalnya, sistem manajemen material yang fungsi intinya adalah manajemen pengadaan dan manajemen gudang. Kelengkapan fungsi intinya menentukan kelas di antara produk yang tersedia di pasar yang memenuhi inti yang sama kegunaan. Dalam hal perangkat lunak manajemen material, dapat berupa versi pengguna tunggal, versi multipengguna yang terbatas pada satu lokasi, atau aplikasi berbasis Web yang melayani lokasi yang berbeda secara geografis. Sedangkan inti fungsionalitas serupa di semua aplikasi ini, kualitasnya berbeda.

Sudut lain untuk melihat kelengkapan fungsionalitas inti produk perangkat lunak adalah volume data yang dapat ditanganinya. Sebuah manajemen materi aplikasi yang dirancang untuk menangani 100.000 item di gudang dan lainnya aplikasi yang dirancang untuk menangani satu juta item memiliki tingkat yang berbeda dari kualitas.

Aspek lain dari fungsionalitas inti adalah fleksibilitas fungsional. Sebuah bahan aplikasi manajemen yang dirancang untuk industri tertentu, seperti rekayasa, misalnya, harus dapat digunakan di setiap jenis industri rekayasa, harus itu industri fabrikasi, permesinan, listrik, atau elektronik. Jika seperti itu kasus, maka perangkat lunak memiliki fleksibilitas fungsional.

Fungsionalitas inti terutama merupakan aspek spesifikasi dan desain. Membandingkan produk perangkat lunak dengan produk pesaing lainnya dan melakukan survei pasar untuk memastikan persepsi pelanggan tentang kesenjangan saat ini produk keduanya merupakan sumber informasi yang berguna untuk meningkatkan fungsi inti dari produk yang sedang dikembangkan. Kualitas fungsionalitas inti dipastikan melalui tinjauan ahli, tinjauan sejawat, pengujian positif, dan pengujian fungsional.

C. FUNGSI TAMBAHAN

Fungsionalitas tambahan adalah fungsionalitas yang melengkapi fungsi inti. Bahkan jika fungsi tambahan tidak ada, produk tetap berguna. AC di dalam mobil adalah fungsi tambahan; sebuah mobil masih berguna meskipun itu tidak memiliki AC. Sekali lagi menggunakan contoh perangkat lunak manajemen material, yang memungkinkan eksekutif mengakses informasi stok, pengadaan status, dll. adalah fungsi tambahan.

Ini adalah fungsi tambahan yang meningkatkan nilai produk perangkat lunak. Fungsi tambahan diklasifikasikan lebih lanjut ke dalam subkategori berikut:

1. Fungsi keselamatan dan keamanan—Ini memberikan keselamatan dan keamanan kepada orang-orang yang menggunakan produk serta produk itu sendiri. Badan mobil, misalnya, melindungi penumpang dari cedera akibat benda terbang. Kaca depan melindungi penumpang dari cuaca dan memberikan visibilitas yang jelas bagi pengemudi. Speedometer memberikan informasi kecepatan untuk memungkinkan pengemudi mempertahankan kecepatan yang dapat diterima. Dalam perangkat lunak manajemen material, mencegah akses yang tidak sah dan membatasi hak modifikasi berdasarkan peran pengguna adalah dua contoh fungsi keselamatan dan keamanan. Fungsionalitas ini dicapai melalui spesifikasi dan desain perangkat lunak. Kualitas fungsi ini dipastikan melalui pengujian keamanan.

2. Fungsi kegunaan—Ini memungkinkan produk digunakan dengan lebih nyaman atau lebih nyaman. Starter otomatis di dalam mobil merupakan peningkatan yang luar biasa dibandingkan dengan menghidupkan mesin. Setir lebih nyaman daripada menggunakan setang untuk mengemudikan mobil. Pengukur bahan bakar melindungi pengemudi agar tidak terdampar karena kehabisan bensin tanpa disadari. Semua fitur ini memfasilitasi penggunaan produk yang lebih nyaman. Dalam perangkat lunak manajemen material, fungsi bantuan online, transfer informasi otomatis dari departemen material ke departemen keuangan, dan warna layar yang dapat disesuaikan merupakan contoh fungsi kegunaan. Contoh lain yang sangat baik dari fungsionalitas kegunaan yang membantu pengguna adalah peringatan yang muncul di layar untuk menunjukkan bahwa tombol caps lock aktif. Fungsionalitas kegunaan dicapai melalui pedoman kegunaan, garis panduan desain, dan pedoman arsitektur perangkat lunak. Kualitas fungsi ini dipastikan melalui ulasan dan pengujian kegunaan.

3. Fungsi toleransi kesalahan—Sedikit menyalahgunakan atau penggunaan yang tidak disengaja tidak akan merusak produk. Kecelakaan kecil seharusnya tidak membuat mobil tidak dapat digunakan. Demikian pula, perangkat lunak harus mentolerir penggunaan yang tidak diinginkan atau penyalahgunaan sampai tingkat yang wajar. Penggunaan perangkat lunak yang tidak disengaja dapat terjadi dalam berbagai cara, dan perangkat lunak harus dapat melindungi dirinya sendiri dari penyalahgunaan ini dan menjaga integritasnya. Fungsionalitas toleransi kesalahan dicapai melalui standar desain, pedoman desain antarmuka pengguna, dan validasi data. Kualitas fungsi ini dipastikan melalui pengujian negatif.

4. Fungsionalitas yang nyaman— Fungsi ini membuat pengguna merasa lebih nyaman saat menggunakan produk. Pendingin udara dan sistem suara di dalam mobil termasuk dalam

kategori ini. Fungsionalitas ini disebut sebagai "lonceng dan peluit" di industri perangkat lunak. Animasi dan grafik khusus adalah contoh dari fungsi ini. Lain adalah pesan kesalahan yang ditunjukkan dengan suara bip. Fungsionalitas yang menyenangkan dicapai melalui pedoman estetika. Kualitas fungsi ini dipastikan melalui tinjauan ahli dan manajerial.

5. Fungsionalitas penghargaan—Fungsi- fungsi ini meningkatkan penampilan produk. Di dalam mobil, sudut dan kontur dalam bentuk bodi, cat khusus, atap yang dapat dilepas, dan jendela otomatis, semuanya merupakan fungsi yang dihargai. Dalam produk perangkat lunak, layar jazzy dengan animasi adalah fungsi penghargaan. Fungsi penghargaan adalahdicapai melalui pedoman desain antarmuka pengguna dan brainstorming. Kualitas fungsi ini dipastikan melalui pengujian positif.

6. Fungsionalitas One-upmanship (competitive edge)—Fungsi ini membantu pengembang perangkat lunak mengalahkan pesaing mereka. Sebuah televisi di dalam mobil adalah fungsi satu keunggulan, seperti telepon dengan kemampuan Internet. Dalam manajemen material, memungkinkan vendor untuk melihat status proposal mereka melalui Internet adalah contoh fungsionalitas satu keunggulan. Fungsionalitas one-upmanship dicapai melalui bimbingan manajerial. Kualitas fungsi ini dipastikan melalui pengujian normal.

Atribut dasar yang dibutuhkan suatu produk untuk mengklaim label kualitas adalah bahwa produk tersebut harus melakukan fungsi yang dirancang untuknya. Jika fungsi-fungsi tersebut dijalankan dengan benar dan hasil yang disampaikan akurat, maka produk tersebut memiliki kualitas dasar, yang paling penting jika produk tersebut dikatakan memiliki kualitas sama sekali.Menggunakan contoh mobil sekali lagi, jika sebuah mobil diharapkan dapat mengangkut empat orang sejauh 500 mil tanpa henti dan memenuhi harapan ini, maka atribut dasar untuk mengklaim label kualitas dipenuhi oleh mobil itu.

D. KOTAK PUTIH (KOTAK KACA) STANDPOINT

Untuk mencapai kualitas produk, produk perangkat lunak dibangun untuk menyertakan karakteristik berikut:

1. Maintainability— Produk dikembangkan sedemikian rupa sehingga dapat dipelihara, artinya fungsionalitas dapat dengan mudah ditambahkan, dipindahkan, atau dimodifikasi. Untuk mencapai pemeliharaan, kode harus dapat dibaca dan dimengerti oleh programmer selain yang asli pengarang. Ini dimungkinkan oleh pembuat kode asli yang mendefinisikan dan mengikuti pedoman pengkodean. Tindakan pencegahan khusus berikut:memungkinkan pemeliharaan kode: Gunakan konvensi penamaan standar sehingga mer program berikutnya dapat membedakan antara variabel program, konstanta, table bidang, dll.

Format kode dengan cara standar sehingga mudah dibaca dan awal dan akhir pernyataan multiline jelas. Gunakan pernyataan dokumentasi sebaris secara ekstensif untuk membantu pemrogram berikutnya memahami logika. Gunakan konstruksi sederhana daripada konstruksi kompleks.

Tabel 3.1. Kualitas produk perangkat lunak dari sudut pandang fungsionalitas

Kegunaan	Bagaimana mencapai	Bagaimana memastikan kualitas?
Inti Kegunaan	Spesifikasi perangkat lunak dan desain perangkat lunak, standar dan pedoman	Pengujian fungsional, ulasan
Tambahan Kegunaan Fungsi keselamatan dan keamanan	Spesifikasi perangkat lunak dan desain perangkat lunak, standard dan pedoman	Pengujian keamanan, ulasan
Fungsi kegunaan	Pedoman kegunaan, pedoman desain, dan pedoman arsitektur perangkat lunak	Ulasan dan kegunaan pengujian
Toleransi kesalahan Tes negative fungsi	Standar desain, pedoman desain antarmuka pengguna, dan validasi data	Tes negatif
Fungsionalitas yang menyenangkan	Pedoman estetika	Tinjauan manajerial
Menghargai fungsi	Panduan desain antarmuka pengguna dan curah pendapat	Tes positif
Fungsionalitas satu keunggulan	Bimbingan manajerial	Tes normal

2. Portabilitas—Portabilitas memungkinkan produk perangkat lunak dipindahkan dari satu platform ke platform lainnya. Untuk mencapai fitur ini, konstruksi standard bahasa pemrograman harus digunakan, dan khusus mesin konstruksi harus dihindari. Aspek ini perlu dimasukkan dalam pedoman pengkodean khusus bahasa.

3. Fleksibilitas—Fleksibilitas adalah fitur utama lain yang harus dimiliki suatu produk jika ingin disebut sebagai “produk berkualitas”. Fleksibilitas membuatnya layak untuk menggunakan produk bahkan jika beberapa nilai berubah, atau memungkinkan produk untuk digunakan di lingkungan yang sedikit berbeda dari yang semula dimaksudkan. Misalnya, jika tarif pajak berubah, kode tidak perlu diubah untuk menerapkan tarif pajak baru. Contoh lain, jika perangkat lunak manajemen persediaan untuk industri manufaktur dapat digunakan di industri farmasi tanpa perubahan kode, maka perangkat lunak tersebut dianggap fleksibel. Fleksibilitas dicapai dengan menghindari hard coding konstanta dan parameterisasi produk sebanyak mungkin. Aspek ini biasanya menjadi bagian dari pedoman pengkodean.

4. Efisiensi—Efisiensi berarti meminimalkan konsumsi sumber daya sistem dan waktu eksekusi. Perangkat lunak menggunakan akses acakmemori dan waktu unit pemrosesan pusat saat sedang beroperasi. Minimisasi dicapai dengan mendeklarasikan variabel secara hati-hati (menghindari deklarasi variabel yang tidak digunakan, menggunakan kembali variabel yang dideklarasikan, dll.), menutup file atau tabel saat penggunaannya selesai, dan tidak mengikat unit pemrosesan pusat dengan periferal yang lambat seperti printer (yang terjadi jika mencetak langsung dari program alih-alih menggulung pekerjaan cetak). Efisiensi dicapai dengan mengikuti bagian garis panduan efisiensi dari pedoman pengkodean.

5. Modularitas—Modularitas mengacu pada pembuatan produk perangkat lunak menggunakan modul yang berdiri sendiri atau hampir berdiri sendiri sedemikian rupa sehingga fungsionalitas tidak terduplikasi, sehingga ketika sebuah modul diubah, perubahan tersebut tidak berdampak pada modul perangkat lunak lainnya. Modularitas dicapai dengan mengikuti pedoman arsitektur perangkat lunak dan pedoman desain perangkat lunak.

6. Reusability—Reusability mengacu pada pengembangan kode sumber sedemikian rupa sehingga dapat digunakan lagi dan lagi di produk lain. Ini dicapai dengan mengembangkan kode sebagai kelas atau sebagai komponen independen sejauh mungkin. Reusability dicapai dengan mengikuti pedoman pengkodean reusability.

7. Keterbacaan— Kode yang dikembangkan harus dapat dibaca, dan keterbacaan adalah syarat pertama untuk pemeliharaan. Hal ini dicapai dengan tepat untuk anyaman kode, memungkinkan pengenalan awal dan akhir dari pernyataan multiline komposit. Keterbacaan dicapai dengan mengikuti bagian pedoman pemformatan dari pedoman pengkodean.

8. Testability—Sebuah produk perangkat lunak dapat diuji ketika setiap produknya unit dapat diuji secara independen. Kode tidak boleh ditulis sedemikian rupa cara bahwa satu set unit perangkat lunak sebelumnya harus dijalankan untuk menguji unit perangkat lunak. Setiap unit perangkat lunak sendiri harus dapat diuji dengan data ujinya. Testability memudahkan untuk memperbaiki cacat selama pengembangan, karena menguji dan memperbaiki cacat satu unit pada satu waktu adalah lebih sederhana daripada menguji satu set unit dan mencoba menemukan asal cacat dari semua unit. Testabilitas dicapai dengan mengikuti pedoman desain perangkat lunak.

Tabel 3.2. Ringkasan fitur kualitas produk dari kotak putih sudut

Fitur	Cara mencapai fitur
Pemeliharaan	Pedoman pengkodean
Portabilitas	Konstruksi standar dan pedoman pengkodean
Fleksibilitas	Pedoman pengkodean dan hindari pengkodean keras dan parameterisasi
Efisiensi	Pedoman efisiensi

Modularitas	Pedoman arsitektur perangkat lunak dan pedoman desain perangkat lunak
Dapat digunakan kembali	Pedoman pengkodean yang dapat digunakan kembali
Keterbacaan	Pedoman pemformatan
Kemampuan untuk diuji	Pedoman desain perangkat lunak

E. ADANYA CACAT PADA PRODUK

Secara umum diterima di seluruh industri—termasuk industri perangkat lunak—bahwa beberapa cacat tetap ada dalam produk bahkan setelah kegiatan jaminan kualitas yang ekstensif telah dilakukan. Meski begitu, setiap organisasi berusaha untuk menghilangkan semua cacat, yang dirancang untuk dicapai oleh aktivitas penjaminan mutu. Sisacacat pada suatu produk seringkali merupakan hasil dari kesenjangan dalam spesifikasi, desain produk, atau pengembangan atau karena pengujian yang tidak memadai. Sebelum melanjutkan, istilah-istilah berikut ini perlu dipahami dengan baik dalam: konteks pengembangan perangkat lunak:

1. Kesalahan atau bug—Kedua kata tersebut memiliki arti yang sama. "Bug" adalah istilah sehari-hari yang digunakan oleh pengembang perangkat lunak, dan proses menghilangkan kesalahan disebut "debugging." Error adalah langkah yang salah dalam sebuah program, definisi data yang tidak tepat (jenis dan ukurannya), atau hasil yang benar yang dihasilkan oleh langkah dalam sebuah program. Kesalahan dimasukkan ke dalam produk selama tahap konstruksi pengembangan perangkat lunak dan telah lolos dari jaring aktivitas jaminan kualitas yang dilakukan pada produk.
2. Cacat — Cacat adalah kondisi yang sudah ada sebelumnya dalam produk jadi. Hal ini dapat disebabkan oleh kesalahan atau lingkungan operasi tempat perangkat lunak digunakan. Cacat dapat berasal dari salah satu dari tiga tahap (spesifikasi, desain, atau pengembangan) dan dapat lolos dari aktivitas jaminan kualitas.
3. Fault— Kesalahan terjadi ketika pengoperasian perangkat lunak mengalami cacat. Kesalahan dan cacat mengintai dalam perangkat lunak sampai bagian (atau fungsi) di mana kesalahan atau cacat berada diakses oleh pengguna. Sebuah kesalahan dapat mengakibatkan kegagalan jika toleransi kesalahan tidak dibangun ke dalam sistem.
4. Toleransi kesalahan—Toleransi kesalahan adalah seperangkat mekanisme yang dibangun ke dalam produk yang memberikan tindakan korektif atau tindakan alternatif sehingga kegagalan tidak terjadi. Dalam produk perangkat lunak, toleransi kesalahan mengacu pada penyediaan tindakan alternatif yang memfasilitasi penggunaan berkelanjutan dari fungsionalitas perangkat lunak lain yang tidak rusak.
5. Kegagalan—Kegagalan adalah hasil dari produk yang mengalami kesalahan saat

   beroperasi di mana fitur toleransi kesalahan tidak ada. Kegagalan mencegah pergantian yang mulus ke fungsi lain. Ini menghentikan pengoperasian produk, dan satu-satunya

cara untuk melanjutkan adalah memulai kembali pengoperasian dari awal. Kegagalan diulangi setiap kali rangkaian kondisi yang sama berulang sampai kesalahan yang menyebabkan kesalahan dihilangkan dari produk.

Seperti yang dinyatakan sebelumnya, tidak layak untuk membangun produk yang 100% bebas cacat. Jadi, pertanyaannya adalah: Cacat mana yang diperbolehkan dan mana yang tidak? Untuk membantu menjawab pertanyaan ini, cacat dapat dibagi menjadi tiga kelas:

1. Cacat kritis —Cacat kritis menyebabkan kegagalan. Mereka tetap dalam produk sampai tindakan korektif diambil untuk menghilangkannya dan produk diperbarui. Cacat ini harus segera diperbaiki.
2. Cacat utama— Cacat utama adalah kondisi kesalahan yang fitur toleransi kesalahannya ada pada produk. Sementara cacat ini tidak mengganggu penggunaan fungsi lain, mereka tetap ada sampai mereka teratasi. Namun, cacat utama bukanlah penghalang; mereka tidak perlu segera diperbaiki, tetapi mereka harus diperbaiki secepat mungkin.
3. Cacat kecil— Cacat kecil hanyalah gangguan. Mereka tidak menyebabkan kegagalan, dan tidak ada tindakan alternatif yang diperlukan. Fungsionalitas dapat digunakan tanpa gangguan, tetapi cacatnya sama saja. Contoh cacat kecil meliputi: Kesalahan ejaan atau ejaan yang salah di layar (Kesalahan ejaan salah di negara mana pun, tetapi ejaan yang salah hanya salah di beberapa negara. Misalnya, "warna" salah di negara yang menggunakan ejaan Inggris, tapi itu benar di negara-negara yang menggunakan ejaan AS.)

Sebuah "produk berkualitas" tidak boleh memiliki cacat kritis. Cacat seperti itu memberi pengguna kesan yang sangat negatif terhadap produk. Konsekuensi dari cacat kritis adalah bahwa pengguna akhir tidak dapat menggunakan fungsionalitas sampai cacat diperbaiki, yang menahan operasi pelanggan. Oleh karena itu, cacat kritis dilarang keras jika suatu produk akan ditandai sebagai "produk berkualitas." Cacat kritis dapat ditemukan melalui tinjauan perangkat lunak, pengujian negatif, dan pengujian stres, di samping jenis pengujian biasa yang dikenakan pada suatu produk.

Cacat kecil membuat produk menjadi bahan tertawaan. Itu terlihat oleh semua orang untuk dilihat, ditunjukkan, dan diolok-olok. Sebuah cacat kecil juga adalah klub yang dengannya label kualitas pengembang produk dapat dikalahkan. Adanya cacat kecil menunjukkan kelemahan kegiatan jaminan kualitas yang dilakukan dalam organisasi. Cacat kecil mudah dijebak dan dihilangkan, dan sebaiknya pengembang perangkat lunak meluangkan waktu untuk menghilangkannya jika mereka ingin mengklaim label kualitas untuk produk mereka. Cacat kecil dapat ditemukan dengan menggunakan sistem verifikasi perangkat lunak dan daftar periksa yang cermat selama peninjauan.

Cacat utama memberikan tindakan alternatif yang menyamarkan cacat dari pandangan pengguna. Selanjutnya, pengguna memahami bahwa serangkaian keadaan tertentu menghasilkan kesalahan. Karena kegagalan bukan akibat dari cacat, pengguna dapat melanjutkan operasi dan biasanya bersedia menunggu agar cacat tersebut dapat diperbaiki. Dengan demikian, cacat utama tidak berbahaya sejauh klaim label kualitas produk. Cacat besar mengintai di dalam produk, dan sangat sulit untuk mengungkap dan menghilangkan semuanya. Mereka muncul hanya di bawah serangkaian kondisi operasi tertentu. Tidaklah praktis untuk menghasilkan setiap kemungkinan kombinasi kondisi operasi selama pengujian perangkat lunak. Itulah sebabnya cacat besar terus mengintai di dalam produk.

Dengan demikian, pengembang perangkat lunak dapat dengan yakin mengklaim bahwa suatu produk bebas cacat selama tidak ada cacat kritis atau kecil dalam produk tersebut. Kehadiran beberapa cacat utama tidak mempengaruhi status label kualitas. Namun demikian, produk dengan kualitas yang sebenarnya tidak boleh mengandung cacat apa pun, karena semua cacat pada akhirnya akan ditemukan oleh pengguna. Satu atau dua cacat mungkin tidak mengganggu pengguna, tetapi terlalu banyak pasti akan menyebabkan pengguna mempertanyakan label kualitas produk.

F. KUALITAS PROGRAM

Program adalah unit terkecil dari perangkat lunak. Selama perubahan terbaru dalam pengembangan perangkat lunak, program mengalami metamorfosis. Program-program dari tahun 1970-an terlihat sangat berbeda dari yang ada sekarang. Program hari ini sekarang disebut sebagai rutinitas, metode, kelas, objek, komponen, agen, dan makro, di antara nama lainnya. Orang mungkin sensitif tentang apa yang disebut program, tetapi faktanya tetap bahwa semua nama ini adalah cara untuk merujuk ke program perangkat lunak.Membangun produk yang berkualitas dimulai dengan unit terkecil dari produk perangkat lunak, dan itu adalah program. Atribut program yang baik adalah sama dengan produk perangkat lunak yang berkualitas (maintainability, flexible, portability, dll.). Atribut-atribut ini dapat dibangun ke dalam program menggunakan pedoman pengkodean umum untuk semua pemrograman dan pedoman pengkodean untuk setiap bahasa pemrograman yang digunakan dalam suatu organisasi. Lampiran I memberikan pedoman pengkodean umum sebagai referensi. Selain informasi yang ditemukan di sana, saran berikut juga berguna dalam mengembangkan program perangkat lunak yang baik. Sebisa mungkin, jangan gabungkan beberapa fungsi menjadi satu program. Simpan satu fungsi untuk satu program. Terkadang satu fungsi dapat menghasilkan program yang lebih panjang, tetapi beberapa fungsi tidak boleh digabungkan hanya karena sebuah program sangat pendek. Hal ini membuat pemeliharaan perangkat lunak lebih mudah.

Jaga agar program tetap pendek. Pada hari-hari pemrograman bahasa generasi ketiga, pemrograman terstruktur adalah norma, dengan satu rutin panggilan utama dan beberapa subrutin yang disebut. Sekarang, bagaimanapun, dengan orientasi acara dalam eksekusi program, sejumlah pemrograman terstruktur dibangun ke dalam bahasa

pemrograman itu sendiri. Oleh karena itu, lebih masuk akal untuk membuat setiap program respons acara singkat. Apa itu program pendek? Tidak ada panjang optimal yang ditentukan secara universal untuk program pendek, tetapi aturan praktis industri adalah bahwa 50 hingga 100 baris adalah program yang dapat dikelola. Kondisi ekstrem mungkin memerlukan program yang lebih lama, tetapi sebisa mungkin, pertahankan setiap rutinitas hingga maksimum 100 baris kode. Manfaatkan secara ekstensif subrutin (subprogram) yang dapat dipanggil alih-alih menulis program yang lebih panjang. Jadikan semua program dan subrutin tujuan umum dengan melewatkan parameter ke mereka. Ini juga memfasilitasi penggunaan kembali kode di proyek lain. Selalu mencoba untuk menggunakan kembali kode yang telah terbukti sedapat mungkin alih-alih menulis kode baru dari awal. Kode yang terbukti telah diuji untuk mengonfirmasi bahwa semua aspek kualitas telah terpasang. Bangun sebanyak mungkin pustaka yang terhubung secara dinamis alih-alih membangun fungsionalitas ke dalam program. Pustaka yang terhubung secara dinamis memasukkan memori akses acak (RAM) sesuai kebutuhan. Ini menghemat RAM dan menghindari penggunaan memori virtual dalam jumlah besar. Hindari pengkodean keras (mendefinisikan konstanta di dalam program itu sendiri) sepenuhnya kecuali dalam kasus flag dan counter untuk loop terbatas. Jika ada banyak konstanta, gunakan file parameter yang dibaca di awal program atau sesuai kebutuhan, dan impor nilainya ke dalam program selama eksekusi.

Gunakan hanya file datar untuk menyimpan parameter program. Jika data parameter ada dalam tabel, beberapa nilai mungkin perlu dikodekan secara keras untuk terhubung ke database dan membuka tabel.

Selalu simpan hanya satu titik masuk dan satu titik keluar untuk program. Beberapa pintu keluar khususnya dapat membuat beberapa tabel, file, atau koneksi terbuka, yang dapat menyebabkan masalah.

Jangan membuka file, tabel, atau koneksi database di awal program; membukanya sebelum digunakan. Ingat bahwa membuka ini menempati minimal satu blok setiap RAM (ukuran blok berubah dari sistem ke sistem, tetapi ukuran normal adalah satu kilobyte). Pada hari-hari ini sistem operasi multiuser dan multitasking, RAM terisi dengan sangat cepat dan sistem mulai menggunakan memori virtual dari disk. Menggunakan memori virtual dalam jumlah besar dapat mengakibatkan thrashing (bertukar terlalu banyak halaman dari RAM ke disk dan sebaliknya), yang menyebabkan komputer menjadi lambat.

Saat membuka file atau tabel atau menghubungkan ke database, pastikan mereka ditutup kembali segera setelah fungsi membaca atau memperbarui selesai. Jangan biarkan file-file ini dan koneksi database ditutup secara otomatis dengan tindakan menutup program.

Saat mendeklarasikan variabel, gunakan sebanyak mungkin variabel lokal (lokal ke rutin di mana mereka digunakan) sebanyak mungkin. Variabel global (statis) tetap aktif meskipun rutin ditutup, dan mereka menempati RAM tanpa perlu sampai produk ditutup.

Deklarasikan satu variabel per baris. Mengapa? Banyak bahasa mengizinkan mendeklarasikan banyak variabel per baris. Ingatlah bahwa setelah program dipromosikan ke produksi, sebagian besar organisasi tidak mengizinkan penghapusan baris kode. Baris yang ada dikomentari dan baris baru dimasukkan sebagai gantinya. Baris komentar ex planatory ditambahkan untuk membuat perubahan ini.

Jika perlu untuk menghapus satu variabel dari daftar variabel yang dideklarasikan pada baris yang sama, baris tersebut perlu dikomentari dan baris baru disisipkan dengan semua variabel lainnya. Proses mengomentari dan menyisipkan baris ini rentan terhadap kesalahan. Semua peluang untuk kesalahan perlu dihilangkan.

Hindari struktur kontrol "goto" kecuali benar-benar penting. Struktur ini mengambil kontrol aliran dari programmer dan dapat menyebabkan beberapa program keluar. Selalu gunakan subrutin yang mengembalikan kontrol ke rutin pemanggilan.

Saat mendefinisikan konvensi penamaan, jangan gunakan nama yang lebih panjang dari 15 karakter, meskipun sebagian besar bahasa mengizinkan penggunaan nama yang lebih panjang untuk variabel.

Nama yang lebih panjang adalah penyebab utama kesalahan sintaks dalam program karena sering terjadi kesalahan ejaan saat mengetik. Hampir setiap kata dapat disingkat menjadi dua atau tiga karakter. Gunakan kata-kata yang disingkat alih-alih kata-kata lengkap untuk membangun nama-nama variabel. Nama variabel 15 karakter dapat mewakili empat kata, di mana masing-masing disingkat menjadi tiga karakter dengan tiga pemisah di antara kata-kata tersebut.

Seberapa sering nama variabel membutuhkan lebih dari empat kata?Sedapat mungkin, jangan mengarahkan output dari program secara langsung ke printer kecuali jika sedang mencetak tanda terima di counter. Output besar khususnya harus diarahkan ke file cetak. Utilitas spooling yang disediakan oleh sistem operasi jauh lebih baik dalam mengontrol printer daripada program yang dikembangkan oleh pengembang perangkat lunak. Utilitas spooling menangani peristiwa seperti pemadaman kertas, kertas macet, printer offline, printer tidak menyala, tinta/toner habis, dll. Kecuali pencetakan harus segera dilakukan, lebih baik menyimpan laporan dan menggunakan program lain yang lebih baik dalam menjalankan fungsi pencetakan. Saat mencetak dari program, pastikan untuk menyertakan rutinitas penanganan kesalahan yang lengkap.

Salah satu penyebab utama perangkat lunak berkualitas buruk adalah menyerahkan penanganan kesalahan ke sistem operasi atau platform pengembangan. Masing-masing dapat mengelola banyak kesalahan, tetapi mereka menyebabkan perangkat lunak menutup secara tiba-tiba, mengakibatkan hilangnya data atau integritas data atau mengharuskan program untuk memulai ulang. Untuk memastikan kelancaran operasi, semua kesalahan harus dijebak, dan pesan yang sesuai harus diakhiri pengguna harus ditampilkan di samping tindakan alternatif yang diizinkan.

Kesalahan berikut harus ditangani di dalam program itu sendiri:

a) Menghubungkan ke database—Periksa apakah koneksi berhasil.

b) Membuka file datar —Periksa keberhasilan dalam membuka file.
c) Membuka tabel—Periksa keberhasilan dalam membuka tabel.
d) Menulis informasi ke file atau tabel—Periksa keberhasilan operasi.
e) Pembagian aritmatika— Penting untuk memeriksa bahwa penyebutnya tidak nol sebelum operasi pembagian aritmatika dilakukan. Jika ya, kesalahan yang tidak dapat dipulihkan akan terjadi.
f) Perkalian aritmatika—Sangat penting untuk memeriksa apakah variabel penerima memiliki lebar yang cukup untuk mengakomodasi hasil tanpa pemotongan.
g) Menghubungkan ke perangkat apa pun—Periksa apakah koneksi berhasil.

Banyak bahasa pemrograman memberikan pernyataan singkat yang dapat digunakan oleh programmer ahli untuk meningkatkan produktivitas kerja pemrograman. Pernyataan-pernyataan ini bagus dan umumnya bekerja dengan sangat baik.

Namun, terkadang pernyataan singkat ini membuat sangat sulit untuk melacak dan men-debug kesalahan, terutama saat mendebug hasil yang tidak akurat. Menulis lebih banyak baris di mana satu baris kompleks akan memungkinkan pengembang untuk melangkah melalui setiap baris dan melacak hasilnya untuk menentukan pernyataan mana yang menyuntikkan kesalahan.

Keuntungan dalam produktivitas pemrograman akan hilang dengan tambahan waktu debugging. Juga penting adalah kenyataan bahwa pengembang tidak dapat memastikan bahwa pemeliharaan perangkat lunak akan dilakukan oleh programmer ahli.

Pemrogram yang bukan ahli mungkin merasa sangat sulit untuk mempertahankan kode tersebut secara efisien. Mengingat pertimbangan ini, lebih baik menulis beberapa baris kode daripada satu baris lanjutan singkat.

Saat ini, sebagian besar bahasa pemrograman mengakomodasi garis yang lebih panjang. Meskipun setiap baris diperbolehkan hingga 255 karakter, tidak mungkin untuk melihat semua 255 karakter dalam satu baris pada layar atau pada cetakan. Ketika garis dililitkan, jeda baris acak, yang membuatnya sulit untuk membaca dan memahami kode. Setiap bahasa pemrograman memungkinkan baris untuk dipecah oleh beberapa mekanisme. Gunakan fasilitas itu, dan pisahkan antrean panjang menjadi beberapa antrean pendek yang nyaman.

Normanya adalah bahwa garis harus terlihat di layar tanpa menggulir penghitungan cakrawala atau garis yang membungkus. Dengan demikian lebih mudah untuk membaca, memahami, dan memelihara kode. Dengan memecah jalur panjang menjadi beberapa jalur pendek, produktivitas meningkat pada tahap tinjauan sejawat dan pemeliharaan selain tahap debugging.

Memecah pernyataan aritmatika panjang menjadi beberapa pernyataan. Debugging pernyataan aritmatika panjang untuk hasil yang tidak akurat bisa sangat membosankan. Pernyataan aritmatika yang pendek dan banyak memudahkan untuk melacak hasil di setiap pernyataan dan menemukan kesalahan.

Gunakan algoritma sederhana sebagai pengganti algoritma kompleks. Skenario tipikal berikut yang pernah saya temui menjelaskan alasannya. Pernyataan perhitungan aritmatika ada dalam sebuah program. Pernyataan berikutnya menaikkan hasilnya ke kuadratnya.

Kemudian pernyataan berikutnya menghitung akar kuadrat dari hasil, dan hasil ini digunakan dalam pernyataan berikutnya. Rupanya, programmer menggunakan algoritme ini untuk menghindari hasilnya menjadi negatif (menaikkan angka apa pun ke kuadratnya membuatnya positif), karena bahasa tersebut tidak menyediakan fungsi nilai absolut. Menemukan akar kuadrat dari suatu angka adalah operasi tumpukan, yang menghabiskan lebih banyak waktu dan sumber daya.

Juga, akar kuadrat dapat menyebabkan masalah presisi, karena jumlah desimal yang panjang. Sebagai gantinya, programmer dapat menggunakan konstruksi if-then dan mengalikan angka dengan minus satu (-1) jika hasilnya negatif (atau kurang dari nol). Untuk sebagian besar masalah kompleks, solusi sederhana memang ada. Gunakan yang lebih sederhana sebagai pengganti algoritma yang kompleks. Solusi sederhana meningkatkan efisiensi pemeliharaan perangkat lunak, jika bukan efisiensi eksekusi.

Dalam beberapa bahasa pemrograman yang tersedia di beberapa sistem, ada ekstensi khusus sistem yang membuat pemrograman lebih mudah. Konstruksi khusus sistem ini juga dapat memfasilitasi pencapaian fungsionalitas yang tidak tersedia dalam bahasa standar.

Namun, menggunakan konstruksi khusus sistem mempengaruhi portabilitas. Oleh karena itu, sejauh mungkin, jangan gunakan konstruksi khusus sistem. Tentukan apakah utilitas perpustakaan yang menggunakan konstruksi standar untuk mencapai fungsionalitas yang sama dapat dikembangkan, dan jika demikian, gunakan di seluruh produk perangkat lunak.

Jika benar-benar diperlukan untuk menggunakan konstruksi khusus sistem, lihat apakah konstruksi tersebut dapat dimasukkan ke dalam rutin yang dapat dipanggil sehingga pengembang mengetahui rutinitas mana yang memerlukan tindakan khusus saat mem-porting perangkat lunak. Gunakan mereka di dalam program utama jika tidak ada alternatif di atas yang bisa diterapkan.

G. PENGUKURAN KUALITAS PRODUK

Pasar menilai kualitas produk komersial, dan penilaian inilah yang menentukan posisi relatif produk di pasar.Produk tunduk pada kekuatan pasar.Untuk produk perangkat lunak yang dibuat berdasarkan pesanan, kualitas dijamin oleh proses pengembangan perangkat lunak dan pengujian penerimaan. Pengujian penerimaan tidak pernah bisa lengkap. Beberapa perusahaan mengukur kinerja proses perangkat lunak, meskipun diamanatkan oleh standar seperti ISO 9000 dan oleh model seperti itu sebagai Integrasi Model Kematangan Kemampuan. Oleh karena itu, kualitas produk perangkat lunak perlu diukur, terutama dalam skenario pengembangan perangkat lunak make-to-order.

Kualitas produk tidak mudah untuk diukur, apalagi untuk pengukuran yang akurat. Isu kedua adalah pada tahap apa kualitas produk perangkat lunak harus diukur. Produk fisik, seperti mobil, menjalani uji coba lapangan aktual atau simulasi yang ekstensif sebelum dirilis ke publik. Produk perangkat lunak yang bersifat komersial juga dikenakan uji coba lapangan, tetapi tidak ada uji coba lapangan yang dilakukan untuk produk perangkat lunak yang dibuat untuk satu pelanggan, kecuali untuk penyediaan dukungan selama masa garansi. Oleh karena itu, jika kualitas produk perangkat lunak akan diukur sama sekali, itu harus dilakukan pada tahap rilis.

Metrik komposit untuk mengukur kualitas produk perangkat lunak, yang didasarkan pada parameter berikut:

1. Lingkungan organisasi yang mendorong kualitas produk
2. Efektivitas kegiatan jaminan kualitas organisasi
3. Cakupan peer review perangkat lunak
4. Cakupan pengujian unit kode
5. Kelengkapan pengujian perangkat lunak

H. RINGKASAN

Fungsionalitas inti mengacu pada fungsionalitas utama yang dirancang untuk dipenuhi oleh produk, yang tanpanya produk tidak berguna. Fungsionalitas tambahan adalah fungsionalitas yang melengkapi fungsi inti. Bahkan jika fungsi tambahan tidak ada, produk tetap berguna. Mengukur kualitas produk perangkat lunak, yang didasarkan pada parameter lingkungan organisasi yang mendorong kualitas produk , efektivitas kegiatan jaminan kualitas organisasi , cakupan peer review perangkat lunak , cakupan pengujian unit kode , kelengkapan pengujian perangkat lunak

I. SOAL LATIHAN

1) Sebutkan tiga cacat di dalam cacat produk
2) Sebutkan dan jelaskan fungsi tambahan yang meningkatkan nilai produk perangkat lunak!
3) Sebutkan 5 metrik komposit dan jelaskan!
4) Sebutkan kesalahan-kesalahan program!
5) Sebutkan karakteristik perangkat lunak!

BAB IV
VERIFIKASI PERANGKAT LUNAK

A. VERIFIKASI

Verifikasi adalah kegiatan yang dilakukan untuk memastikan bahwa sesuatu sesuai dengan spesifikasi yang terdokumentasi, standar, peraturan, dan lain-lain.

Tujuan verifikasi terutama untuk memastikan bahwa hal yang benar adalah dibangun, dan konfirmasi spesifik berikut dicari dari verifikasi:

1. Pencapaian fungsionalitas inti yang ditetapkan untuk artefak—Ini memastikan bahwa artefak berisi semua fungsionalitas terkait produk yang seharusnya dan tidak ada fungsi tertentu yang ditinggalkan.

2. Kelengkapan dan kelengkapan artifak—Hal ini memastikan bahwa setiap aspek yang diperlukan dalam artifak tidak ketinggalan. Semua bagian diisi. Semua penjelasan diberikan. Semua aspek artefak mandiri dan semua referensi diberikan dengan benar ke artefak eksternal. Ada ketertelusuran untuk semua aspek ke artefak hulu dan artefak hilir.

3. Kesesuaian dengan standar dan pedoman yang ditetapkan untuk jenis artefak dalam rencana proyek—Ada standar dan pedoman untuk dokumentasi, pengkodean, konvensi penamaan, dll. untuk semua artefak perangkat lunak. Standar dan pedoman ini memastikan keseragaman dalam interpretasi, kelengkapan artefak, dan pemeliharaan. Aspek ini memastikan bahwa artefak sesuai dengan standar dan pedoman ini.

4. Efisiensi dan efektivitas solusi yang disajikan oleh arti fakta—Aspek efisiensi memastikan bahwa alternatif solusi yang dipilih paling sesuai dengan situasi dalam hal kemampuan untuk mengimplementasikan, pemeliharaan, akurasi, dan waktu respons. Aspek efektivitas memastikan bahwa alternatif solusi yang dipilih memberikan hasil yang diharapkan darinya.

5. Kejelasan dan kebenaran artifak— Aspek ini memastikan bahwa artifak tersebut mudah dipahami oleh orang lain, karena mungkin memerlukan perawatan di kemudian hari. Artinya, artefak tidak terbuka untuk multitafsir. Ini juga memastikan bahwa apa yang terkandung dalam artefak itu benar dan tidak ada kesalahan yang masuk.

6. Pencapaian semua fungsi tambahan yang diharapkan dari artefak— Artefak kemungkinan berisi fungsionalitas tambahan yang diperlukan untuk pencegahan cacat, perlindungan terhadap penyalahgunaan atau penggunaan yang tidak diinginkan, fungsi keamanan untuk membatasi hak akses, dll.

Peninjau mengonfirmasi bahwa semua fungsi tambahan terpasang dengan benar ke dalam artefak.

7. Tidak ada fungsi yang tidak perlu dalam artefak—Peninjau juga perlu mengonfirmasi bahwa artefak hanya berisi fungsi yang diperlukan dan tidak ada fungsi lain, baik yang tidak berbahaya atau tidak.

8. Tidak ada fungsionalitas berbahaya yang terkandung dalam artefak—Ini , sekali lagi, fungsionalitas yang tidak perlu, tetapi fungsionalitas yang dapat berbahaya dengan potensi menyebabkan kerusakan. Peninjau perlu berhatihati untuk memastikan bahwa tidak ada fungsi jahat yang menyusup ke dalam artefak.

9. Format artefak mematuhi standar organisasi atau spesifikasi pelanggan—Terkadang pelanggan menentukan formatnya, terutama untuk artefak informasi seperti dokumen desain. Ketika tidak ada spesifikasi pelanggan seperti itu yang diamanatkan, standar organisasi menjadi operatif. Pemformatan yang tepat membantu dalam pemeliharaan.Oleh karena itu, peninjau perlu memastikan bahwa artefak diformat dengan benar. Seringkali, pemformatan kode diabaikan dalam artefak kode, yang menyebabkan masalah selama pemeliharaan perangkat lunak. Peninjau perlu memastikan bahwa pemformatan dilakukan dengan benar dan sesuai dengan standar yang ditentukan.

10. Pemenuhan persyaratan fungsi hilir artefak ketika akan digunakan di hilir—Hal ini terutama penting untuk artefak informasi seperti dokumen persyaratan, dokumen desain, dokumen kasus uji, dll. yang akan digunakan oleh orang lain hilir untuk melakukan pekerjaan selanjutnya. Artifak semacam itu perlu diverifikasi untuk memastikan orang lain dapat memahaminya dalam konteks yang tepat tanpa salah tafsir dan dapat melaksanakan pekerjaannya secara efektif dan efisien.

Alat dan teknik untuk verifikasi yang diadopsi dalam pengembangan perangkat lunak organisasi antara lain sebagai berikut:

1. Walkthroughs (also referred to as peer reviews)

2. Inspections

3. Audits

Masing-masing metode verifikasi ini dibahas secara lebih rinci di bagian berikut.

B. WALKTHROUGHS (PEER REVIEWS)

Walkthrough adalah aktivitas di mana orang selain penulis menelusuri setiap kalimat artefak informasi perangkat lunak (terutama dokumen) dan setiap baris kode untuk artefak kode perangkat lunak (kode sumber, skrip tabel, prosedur tersimpan, rutinitas antarmuka, dll. .). Dokumen model CMMI® mendefinisikan peer review di bagian glosariumnya sebagai “review produk kerja yang dilakukan oleh rekan-rekan selama pengembangan produk kerja untuk mengidentifikasi cacat untuk dihapus.

Orang-orang (peer) yang melakukan walkthrough biasanya memiliki pengalaman dan keahlian yang sama dengan penulis. Mereka menyampaikan laporan di akhir panduan, yang dikenal sebagai laporan tinjauan. Laporan tinjauan berisi informasi berikut:

1. Cacat yang ditemukan selama penelusuran, yang mencakup kesalahan logika, kode sampah dan kode berbahaya, variabel dan konstanta yang tidak digunakan tetapi dinyatakan, adanya pengkodean keras, ketidaksesuaian dengan standar dan pedoman, dll.

2. Peluang untuk perbaikan, jika ada, ditemukan selama penelusuran, seperti konstruksi yang lebih baik, peningkatan sintaksis, dll.

3. Saran untuk perbaikan yang mungkin menghasilkan kejelasan atau ketahanan yang lebih baik atau efisiensi eksekusi yang lebih baik, menggunakan komponen yang dapat digunakan kembali, menghilangkan redundansi kode pada artefak atau produk akhir artefak, dll.

4. Informasi lain, seperti nama artefak yang direview, tanggal review, nama penulis dan reviewer, informasi yang berkaitan dengan penutupan cacat, dan tempat untuk memberikan jawaban atas peluang perbaikan serta saran untuk perbaikan

Ada lima jenis walkthrough:

1. Panduan Independen

2. Panduan Terpandu

3. Panduan Grup

4. Tinjauan Ahli

5. Tinjauan Manajerial

Masing-masing dibahas secara lebih rinci di bagian berikut.

C. PANDUAN INDEPENDEN

Jenis verifikasi ini juga terkadang disebut sebagai tinjauan pos. Ini dilakukan dengan cara berikut:

Review Report

Project name:

Name of the artifact being reviewed, with version number:

Name(s) of the reviewer(s):

Name of the author of the artifact:

Date(s) on which the review is conducted:

Type of review: ☐ Independent/guided
☐ Individual/group
☐ Postal/meeting

Defects uncovered during the audit (use an additional sheet if necessary)

Defect ID	Defect description	Reference to process for the defect	Defect origin	Closed on	Status (open or closed)

Signature of lead reviewer:

Date of signature:

Gambar 4.1 . Tinjau format laporan (halaman 1 dari 2)

Closure Action by the Author

Corrective actions implemented

Corrective action implemented	Defect IDs covered by this corrective action	Comments

Preventive action implemented

Preventive action implemented	Defect IDs covered by this corrective action	Comments

Signature of author:

Date of signature:

Defect closure actions (to be filled in by the lead reviewer)
I have verified and found that all the defects described above are closed satisfactorily, except the following defects, which are retracted or pending:

1.

2.

3.

Signature of lead reviewer:

Date of signature:

Gambar 4.2 . Tinjau format laporan (halaman 2 dari 2)

1. Penulis artefak menyelesaikan pekerjaan dan memberi tahu pemimpin proyek (PL) atau manajer proyek perangkat lunak (SPM), yang mengatur peninjauan.

2. PL atau SPM membawa artefak ke dalam kontrol konfigurasi.

3. PL atau SPM mengalokasikan pekerjaan peninjauan artefak kepada rekan penulis artefak.

4. Artefak yang akan direview tersedia untuk reviewer. Ini dapat dilakukan dengan menunjukkan lokasi artefak kepada peninjau atau dengan mentransfer artefak secara fisik kepada peninjau. Ketika lokasi ditunjukkan, hak akses terbatas pada "hanya baca."

5. Reviewer meninjau artefak, menyiapkan laporan review, dan menyerahkan laporan review ke PL atau SPM.

6. PL atau SPM memeriksa laporan review dan mengaturcacat yang ditunjukkan dalam laporan untuk diperbaiki.

7. PL atau SPM meminta agar peninjau memverifikasi kemanjuran cacat yang diperbaiki.

8. Peninjau memverifikasi perbaikan.

9. Jika semua cacat ditutup, peninjau mencatat detailnya meninjau laporan dan menutup laporan.

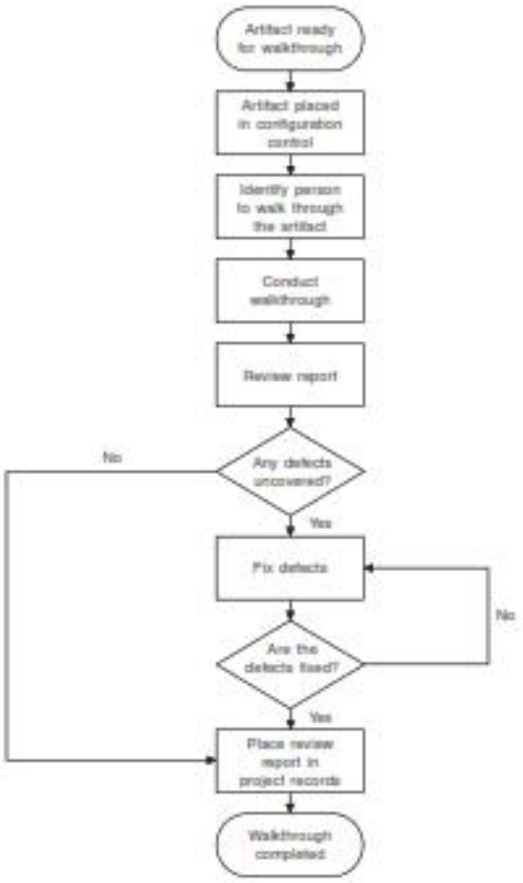
10. Jika cacat tetap ada pada artefak, langkah 6, 7, dan 8 diulang sampai semua cacat diperbaiki.

11. Ketika semua cacat ditutup, aktivitas peninjauan untuk artefak adalah lengkap.

Aspek penting dari tinjauan ini adalah bahwa pembuat artefak tidak perlu hadir saat tinjauan sedang dilakukan. Jika peninjau membutuhkan klarifikasi, dia menghubungi penulis untuk mendapatkan klarifikasi yang diperlukan. Dengan cara ini, penulis bebas untuk mencurahkan waktu untuk aktivitas lain saat artefak sedang ditinjau. Namun, jika artefak tidak berkembang dengan baik atau jika menggunakan algoritme yang kompleks, peninjau mungkin tidak memahami dan mungkin tidak dapat menemukan semua cacat secara efektif.

D. PANDUAN TERPANDU

Panduan dipandu dilakukan di hadapan penulis artefak yang sedang ditinjau. Dalam metode ini, penulis artefak memandu peninjau melalui artefak.



Gambar 4.3 Proses penelusuran independen

Sebuah walkthrough dipandu dilakukan dengan cara berikut:

1. Penulis artefak menyelesaikan pekerjaan dan memberi tahu PL atau SPM, yang mengatur peninjauan.

2. PL atau SPM mengalokasikan pekerjaan peninjauan artefak kepada rekan penulis artefak.

3. Penulis berinteraksi dengan reviewer dan mereka menyetujui slot waktu untuk review.

4. Penulis kemudian memandu resensi melalui artefak, menjelaskan isinya.

5. Reviewer mencari penjelasan tambahan jika diperlukan.

6. Di mana pun peninjau menemukan peluang untuk perbaikan, penulis dan peninjau membahas peluang tersebut, dan jika konsensus tercapai, penulis membuat catatan koreksi yang akan dilakukan keluar.

7. Pada akhir panduan, penulis akan mencatat semua peluang untuk perbaikan dan menerima cara perbaikan yang akan dilakukan.

8. Opsional, penulis dapat menerapkan koreksi pada saat review, menutup setiap cacat seperti yang terungkap.

9. Penulis menerapkan koreksi yang diterima dan menutup kekurangannya sependapat dengan pengulas.

10. Reviewer menginformasikan kepada PL atau SPM bahwa artefak tersebut telah lulus tinjauan.

11. PL atau SPM membawa artefak di bawah kendali konfigurasi.

12. Secara opsional, peninjau dapat menyiapkan laporan tinjauan formal.

Keuntungan dari metode ini adalah durasi peninjauan jauh lebih singkat dibandingkan dengan penelusuran independen. Total waktu penyelesaian untuk penyelesaian tinjauan juga lebih singkat. Satu kelemahan, bagaimanapun, adalah bahwa ada kemungkinan bahwa penulis dan peninjau mungkin tidak setuju tentang peluang untuk perbaikan, yang dapat berakhir dengan argumen yang memerlukan resolusi dari tingkat otoritas yang lebih tinggi. Kerugian lain adalah bahwa penulis mungkin meyakinkan pengulas bahwa cacat sebenarnya bukan cacat!

E. PANDUAN GRUP (ULASAN GRUP)

Panduan kelompok (juga disebut sebagai tinjauan kelompok) digunakan ketika ditentukan bahwa pengetahuan lebih dari satu orang diperlukan untuk meninjau artefak. Jenis penelusuran ini digunakan khususnya untuk artefak strategis, seperti dokumen spesifikasi produk, dokumen desain arsitektur perangkat lunak, dan rencana strategi pengujian.

Panduan grup dilakukan dalam salah satu dari tiga mode berikut:

Tinjauan pos—Dalam metode ini, PL atau SPM bertanggung jawab untuk mengoordinasikan tinjauan. Keuntungan dari jenis ulasan ini adalah bahwa pengulas dapat melakukannya di kenyamanan dan lokasi merekadaripada mencoba mencari waktu di mana pengulas dapat bertemu, yang dapat menunda peninjauan. Selain itu, dengan setiap anggota kelompok peninjau berfokus pada artefak, pengetahuan dari berbagai bidang dapat

menjelaskan lebih banyak tentang kemungkinan peningkatan, yang mungkin tidak dapat dilakukan jika hanya satu orang yang meninjau artefak.

Kerugiannya, bagaimanapun, adalah bahwa peninjauan mungkin memakan waktu lebih lama, karena prosesnya tidak dapat diselesaikan sampai anggota yang paling lambat selesai. Langkah-langkah berikut dilakukan dalam jenis tinjauan ini: 1. Ketika sebuah artefak dibangun dan siap untuk ditinjau, PL atau SPM memilih tim peninjau yang terdiri dari anggota yang sedikit lebih berpengalaman daripada penulis.

2. Koordinator peninjau dinominasikan baik oleh tim peninjau itu sendiri atau oleh PL atau SPM. Terkadang PL atau SPM bertindak sebagai koordinator peninjauan.

3. Artefak diserahkan kepada setiap anggota tim peninjau.

4. Tim peninjau berjalan melalui artefak, dan setiap anggota menyerahkan laporan penelaahan masing-masing kepada koordinator peninjau.

5. Koordinator review mengkonsolidasikan laporan review dengan menghilangkan duplikat dan menyiapkan laporan review akhir.

6. Sebagai alternatif, koordinator tinjauan dapat mengatur pertemuan untuk menyusun temuan tinjauan dan menyiapkan laporan tinjauan akhir.

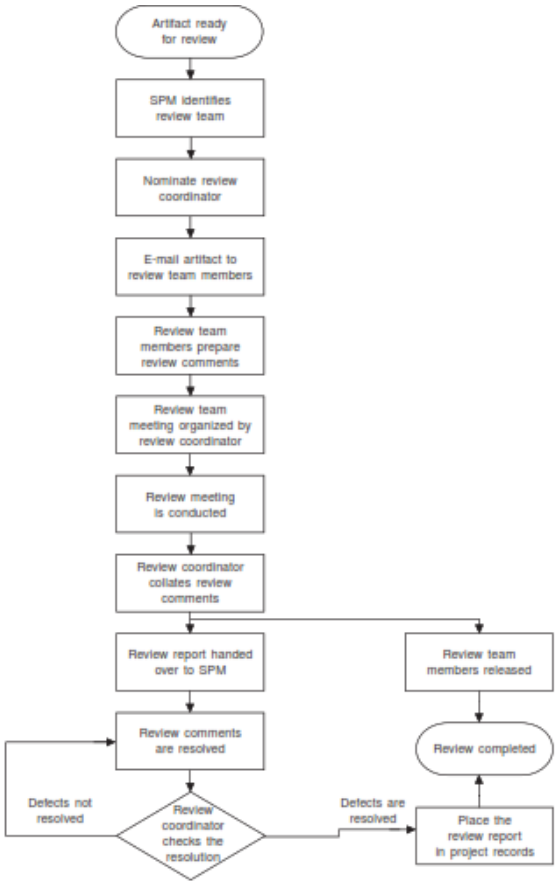
7. Laporan tinjauan akhir diserahkan kepada penulisartefak.

8. Penulis berinteraksi dengan koordinator tinjauan, memperoleh klarifikasi jika diperlukan, dan menerapkan koreksi berdasarkan semua temuan laporan tinjauan.

9. Penulis berinteraksi dengan koordinator peninjau dan menutup semuatemuan laporan tinjauan.

10. Peninjauan selesai dan artefak dipromosikan ke yang berikutnyatahap dalam manajemen konfigurasi.

Proses review pos digambarkan pada Gambar 4.4.



Gambar 4.4 Proses peninjauan pos

Tinjauan rapat—Tinjauan rapat digunakan untuk mengurangi waktu penyelesaian tinjauan. Satu kelemahan, bagaimanapun, adalah bahwa, kecuali koordinator tinjauan ulang, beberapa anggota kelompok peninjau mungkin tidak memusatkan perhatian yang memadai pada artefak, sehingga

tidak berkontribusi sepenuhnya untuk perbaikannya. Langkah-langkah berikut dilakukan dalam jenis tinjauan ini:

1. Ketika sebuah artefak dibangun dan siap untuk ditinjau, PL atau SPM memilih tim peninjau yang terdiri dari anggota yang sedikit lebih berpengalaman daripada penulis.

2. Koordinator peninjau dinominasikan oleh PL atau SPM.

3. Artefak diserahkan kepada setiap anggota tim peninjau sebelum pertemuan.

4. Koordinator review mengatur pertemuan review dengan berkonsultasi dengan semua anggota tim review.

5. Semua anggota tim peninjau datang ke pertemuan dengan persiapan komentar ulasan mereka.

6. Koordinator review mengumpulkan komentar review dari semua anggota kelompok, dan dibahas dalam pertemuan.

7. Laporan review diselesaikan dalam rapat.

8. Anggota tim peninjau dibebaskan, dan koordinator peninjau melanjutkan langkah selanjutnya dalam proses.

9. Koordinator peninjau menyerahkan laporan kepada PL atau SPM, yang meminta pembuat artefak memperbaiki cacat.

10. Penulis berinteraksi dengan koordinator peninjau sampai semua defects diperbaiki dan ditutup oleh koordinator review.

11. Artefak dipindahkan ke kontrol konfigurasi.

Tinjauan rapat terpandu— Jenis tinjauan ini dilakukan dengan cara yang sama seperti tinjauan rapat, kecuali anggota tim tidak harus datang ke rapat dengan menyiapkan komentar. Artefak disajikan oleh penulisnya, dan anggota tim peninjau mendiskusikan setiap topik dan memberikan komentar mereka. Koordinator review menyusun komentar dan menyiapkan laporan review.

Keuntungan dari metode ini adalah waktu penyelesaian dikurangi menjadi waktu sesingkat mungkin dan anggota tim peninjau tidak perlu menghabiskan waktu membaca artefak dan memberikan komentar mereka sebelum datang ke rapat. Namun, satu kelemahannya adalah bahwa anggota tim peninjau mungkin tidak memfokuskan perhatian yang memadai untuk menambah nilai pada artefak.

F. ULASAN AHLI

Kadang-kadang tidak ada seorang pun di tim proyek atau dalam organisasi yang memiliki pengetahuan dan pengalaman dalam bidang atau teknologi di mana artefak itu dibangun.

Ini terutama benar ketika artefak melibatkan teknologi baru. Artefak dapat berupa program yang dikembangkan dalam bahasa baru (atau bahasa baru bagi organisasi) yang hanya dipelajari dan dikuasai oleh penulis, atau organisasi mungkin bekerja dengan domain untuk pertama kalinya.

Selalu ada waktu pertama untuk bekerja dengan setiap teknologi atau domain untuk sebuah organisasi. Dalam beberapa proyek, mungkin perlu untuk memasukkan algoritma matematika yang kompleks di beberapa artefak. Dalam hal ini, hanya algoritma kompleks yang mungkin membutuhkan ahli matematika. Sisanya dapat ditinjau dalam proyek tim.

Dalam proyek besar, arsitektur perangkat lunak memainkan peran penting dalam ketahanan produk. Setelah dibangun dengan arsitektur tertentu, tidak mungkin untuk memperbaiki produk tanpa perombakan besar-besaran. Fakta ini menjadi semakin penting ketika mengembangkan produk komersial, karena arsitektur perangkat lunak yang tidak efisien dapat menyebabkan malapetaka bagi keberhasilan produk. Dalam kasus seperti itu, sebaiknya arsitektur perangkat lunak (atau desain tingkat tinggi) ditinjau oleh para ahli di luar proyek.

Ketika situasi seperti itu terjadi, tinjauan menjadi penting untuk memastikan kelengkapan dan efektivitas artefak. Untuk memastikan bahwa artefak ditinjau secara efektif, seorang ahli baik dari dalam organisasi maupun dari luar organisasi diidentifikasi untuk melakukan tinjauan. Beberapa organisasi (terutama yang besar) memiliki pusat keunggulan atau kumpulan ahli dari mana sumber daya ditarik untuk membantu tim proyek pada saat kesulitan proyek. Jika tidak ada pakar internal, pakar eksternal, mungkin dari institusi akademis, dapat dilibatkan untuk melakukan tinjauan.

Organisasi menggunakan pakar untuk mengembangkan aplikasi serta memverifikasinya.

Pakar dapat berupa salah satu dari jenis berikut:

1. Pakar domain —Orang yang memiliki pengalaman bertahun-tahun dalam lapangan dan yang telah melihat semua kemungkinan situasi.

2. Ahli materi pelajaran —Orang yang ahli dalam mata pelajaran tertentu, seperti matematika, dan biasanya berasal dari akademisi. Para ahli ini mungkin tidak berpengalaman di lapangan, tetapi mereka berpengalaman dalam teori dan dapat membantu dalam pengembangan atau peninjauan algoritma.

3. Pakar teknologi —Orang yang sangat ahli dalam platform pengembangan. Mereka mungkin memiliki pengalaman bertahun-tahun dalam bahasa pemrograman, database, atau platform target tempat produk berfungsi.

4. Pakar sosial —Orang yang memiliki keahlian di bidang perilaku sosial, kekuatan pasar, dan antropologi. Mereka membantu organisasi dalam penerimaan produk oleh pasar sasaran serta

dalam aspek kegunaan produk dan kemungkinan dampak sosialnya. Para ahli ini dapat membantu dengan produk seperti game dan aplikasi multimedia.

Tinjauan ahli mahal, karena ahli berada di luar proyek dan dibayar untuk keahliannya. Tinjauan ahli harus dijadwalkan dengan hati-hati sehingga dapat memanfaatkan waktu ahli secara efektif dan penuh. Jika tinjauan pos (peninjauan independen) digunakan, ahli mungkin memerlukan lebih banyak waktu untuk memahami artefak dan akan mengenakan biaya untuk waktu itu. Namun, ahli mungkin memerlukan klarifikasi dari penulis, dan oleh karena itu interaksi antara penulis dan peninjau ahli tidak dapat dihindari. Untuk alasan ini, panduan terpandu adalah mode tinjauan biasa untuk melakukan tinjauan ahli.

Tinjauan ahli dapat melibatkan satu ahli atau tim ahli, tergantung pada artefak yang ditinjau. Khususnya dalam hal spesifikasi produk atau dokumen arsitektur perangkat lunak dalam domain baru atau teknologi baru, adalah normal untuk menggunakan banyak pakar. Dalam kasus lain, satu ahli saja sudah cukup.

Biasanya, tinjauan ahli dilakukan untuk melengkapi tinjauan sejawat. Tinjauan sejawat yang terperinci dapat dilakukan selain tinjauan ahli untuk memastikan bahwa semua cacat yang tersembunyi ditemukan dan diperbaiki. Metodologi tinjauan pakar serupa dengan penelusuran independen dan penelusuran terpandu, kecuali bahwa peninjau adalah pakar yang dipilih. Semua aspek lainnya sama.

G. ULASAN MANAJERIAL

Tinjauan manajerial dilakukan oleh orang yang secara langsung mengawasi penulis artefak. Tinjauan ini adalah langkah terakhir sebelum mempromosikan artefak ke tingkat berikutnya dalam manajemen konfigurasi. Tinjauan manajerial adalah tinjauan sepintas, dan tidak menyelidiki detail artefak.

Tujuan dari tinjauan manajerial meliputi hal-hal berikut:

1. Itu mengumpulkan artefak untuk memastikan bahwa produk yang tepat dibuat.

2. Menggunakan firasat yang dikembangkan dengan baik dari supervisor, secara khusus melihat area masalah yang mungkin dan memastikan bahwa semuanya baik-baik saja.

3. Ini memastikan bahwa tidak ada informasi yang diperlukan yang hilang dari artefak.

4. Ini memastikan bahwa semua aktivitas penting sebelumnya telah dilakukan sebelum sesuai dengan persetujuan artefak atau mempromosikan artefak ke tingkat berikutnya dalam manajemen konfigurasi.

5. Ini memastikan konsistensi dengan artefak lain dalam proyek serta dalam organisasi.

6. Ini juga memastikan ketertelusuran baik dengan hulu maupun hilir tifak, jika berlaku.Tidak ada aturan keras dan cepat yang melarang supervisor menggali detail artefak, dan terkadang mereka melakukan hal itu, terutama ketika individu yang melakukan peer review sedikit tidak berpengalaman dalam proses review.

Tinjauan manajerial dimulai setelah tinjauan sejawat dilakukan dan semua cacat diperbaiki dan ditutup. Tinjauan manajerial dapat berupa penelusuran independen atau penelusuran terpandu, tergantung pada ukuran dan jenis artefak.

Biasanya, tinjauan manajerial tidak menghasilkan laporan tinjauan. Jika cacat ditemukan selama tinjauan manajerial, pekerjaan tersebut dipindahkan ke orang yang sama (atau tim) yang melakukan tinjauan sejawat atau ke orang lain untuk mengulang tinjauan sejawat. Tinjauan manajerial tidak diharapkan untuk mengungkap cacat apa pun, tetapi jika cacat ditemukan, maka tinjauan sejawat putaran lain dilakukan.

H. PRAKTIK TERBAIK DALAM WALKHTHROUGH

Perangkap pertama dan paling umum yang saya lihat banyak organisasi menjadi mangsa adalah memperlakukan penelusuran sebagai formalitas belaka untuk membuat catatan tinjauan daripada menggunakannya untuk meningkatkan kualitas produk. Organisasi semacam itu kebanyakan menggunakan walkthrough independen atau dipandu yang dilakukan oleh rekan. Penelusuran kelompok dan tinjauan ahli jarang dilakukan, jika pernah. Praktik terbaik adalah memperlakukan penelusuran sebagai alat yang efektif untuk mengungkap sebanyak mungkin cacat dan memberikan aktivitas tersebut pentingnya dan keseriusan yang layak dalam organisasi.

Selain itu, jika perlu, penelusuran kelompok dan penelusuran ahli harus digunakan dalam suatu organisasi. Perangkap kedua yang saya lihat adalah penghilangan penelusuran grup bahkan untuk artefak yang paling penting, seperti dokumen persyaratan dan dokumen desain perangkat lunak. Praktik terbaik adalah memberikan artefak penting pada penelusuran kelompok dan tinjauan ahli.

Perangkap lain yang saya amati adalah tidak melakukan tinjauan sejawat dan tinjauan manajerial. Organisasi dapat membuat artefak untuk tinjauan sejawat diikuti oleh tinjauan manajerial sepintas, jika ada sama sekali, atau benar-benar melewatkan tinjauan sejawat dan melakukan tinjauan manajerial menyeluruh. Masing-masing tinjauan ulang ini memiliki tujuannya sendiri-sendiri. Sementara tinjauan sejawat hanya berfokus pada artefak, tinjauan manajerial memperlakukan artefak sebagai bagian dari keseluruhan rangkaian artefak dan menghubungkan informasi yang terkandung dalam artefak yang ditinjau dengan informasi yang terkandung dalam artefak lain. Sementara tinjauan sejawat berfokus pada detail, tinjauan manajerial berfokus pada gambaran besar. Oleh karena itu, praktik terbaik adalah melewatkan baik tinjauan sejawat maupun tinjauan manajerial.

Di sebagian besar organisasi saat ini, pelaporan cacat diotomatisasi melalui alat perangkat lunak resolusi cacat. Beberapa organisasi menggunakan mekanisme ini hanya untuk melaporkan cacat yang ditemukan selama penelusuran, dan mereka mengabaikan laporan tinjauan secara keseluruhan. Ini adalah perangkap lain. Bahkan jika cacat dilaporkan menggunakan alat perangkat lunak manajer cacat, laporan tinjauan masih perlu disiapkan dan disampaikan. Laporan tinjauan berisi informasi selain mengidentifikasi cacat, seperti peluang dan saran untuk perbaikan. Jika hanya alat pengelola cacat yang digunakan, peninjau tidak disarankan untuk

melakukan pengamatan tentang kemungkinan perbaikan. Oleh karena itu, praktik terbaik adalah menyiapkan laporan tinjauan untuk setiap penelusuran yang dilakukan di organisasi, apakah perangkat lunak pengelola cacat digunakan untuk melacak dan menutup semua cacat atau tidak.

I. INSPEKSI

Inspeksi memainkan peran penting dalam QA perangkat lunak. Mereka memastikan bahwa semua komponen yang diperlukan siap untuk tahap berikutnya. Output dari aktivitas inspeksi adalah laporan inspeksi yang menentukan apakah sistem lulus atau gagal.

Gambar 5.4 menawarkan format yang disarankan untuk laporan inspeksi.

Inspection Report

Project ID:

Type of inspection carried out: ☐ Readiness for system testing
☐ Acceptance testing
☐ Delivery

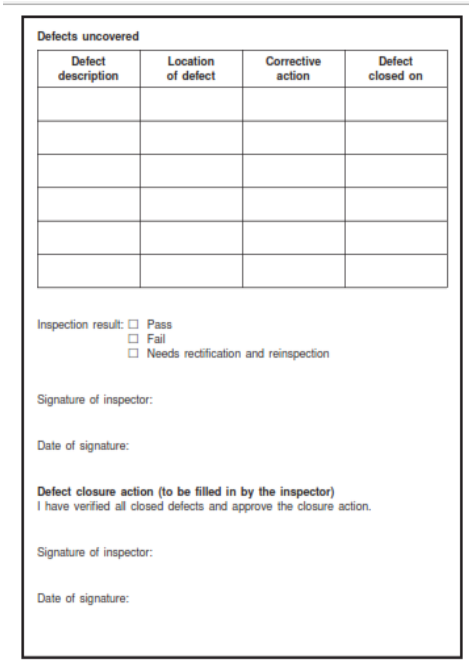
Name of inspector:

Date of inspection:

List of components inspected

Name of component	Nature of component	Type of inspection carried out
Database server	Hardware	Visual and power-on inspection
	RDBMS	Checked for existence of database and tables
	Master data	Checked for existence of data
Software components	Software	Tallied with configuration register Verified all QA records and ensured that all planned QA activities are performed and all defects uncovered are closed
Web server	Hardware	Visual and power-on inspection
	Web server software	Checked for existence of Web server software as well as the Web site
User documentation	Documentation	Verified review records and approvals

Gambar 4.5 Format Laporan Inspeksi (Halaman 1 dari 2)



Defects uncovered

Defect description	Location of defect	Corrective action	Defect closed on

Inspection result: ☐ Pass
☐ Fail
☐ Needs rectification and reinspection

Signature of inspector:

Date of signature:

Defect closure action (to be filled in by the inspector)
I have verified all closed defects and approve the closure action.

Signature of inspector:

Date of signature:

Gambar 4.6 Format Laporan Inspeksi (Halaman 2 dari 2)

Jika laporan menunjukkan sistem lulus pemeriksaan, artinya sistem dapat melanjutkan ke langkah berikutnya. Jika laporan inspeksi menunjukkan bahwa sistem gagal, itu berarti bahwa beberapa perbaikan perlu dilakukan pada sistem yang sedang diperiksa.Setelah perbaikan, sistem dikirim ulang untuk diperiksa. Untuk dapat melanjutkan ke langkah berikutnya, laporan inspeksi harus menunjukkan bahwa sistem lulus inspeksi.

Gambar 4.7 menggambarkan proses pemeriksaan.

Skenario di mana inspeksi akan dilakukan dibahas dalam bagian berikut.

J. PEMERIKSAAN KESIAPAN PENGUJIAN SISTEM

Kesiapan pengujian sistem memastikan bahwa semua komponen yang diperlukan tersedia untuk melakukan pengujian sistem. Khususnya pada aplikasi berbasis Web, banyak konfigurasi diperlukan untuk pengujian sistem yang menyeluruh. Jika salah satu komponen yang diperlukan hilang, proses pengujian akan terhenti dan cacat harus segera diperbaiki.

Keterlambatan dalam memperbaiki cacat membuat penguji menganggur dan menunggu.Pemeriksaan kesiapan pengujian sistem memastikan bahwa server dilengkapi dengan hal-hal berikut:

1. Perangkat Keras (Web server, database server, app server), yang memiliki RAM yang memadai, kapasitas hard disk, dan perangkat keras jaringan

2. Perangkat lunak sistem, termasuk sistem operasi dan perangkat lunak lainnya,yang dimuat dan dalam urutan kerja

3. Perangkat lunak keamanan, seperti antivirus, antispyware, firewall, dll, yang dimuat dan berfungsi dengan baik

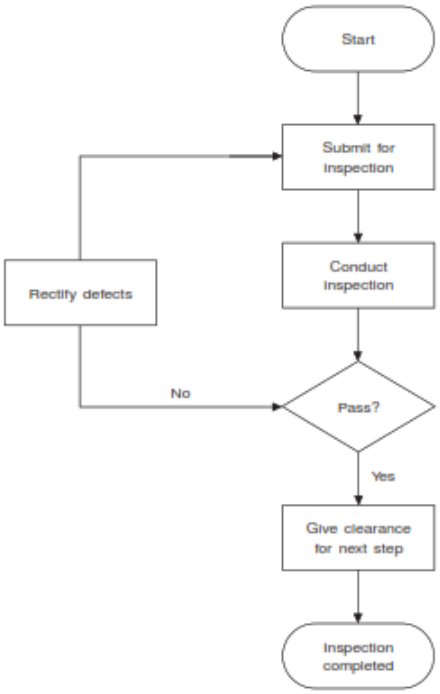
4. Konektivitas dan bandwidth, artinya server terhubung bersama dan terhubung ke mesin klien melalui Internet atau intranet

5. Data master, artinya tabel yang menyimpan data master diisi dengan data yang sesuai dan telah dilakukan kegiatan QA yang diperlukan untuk memastikan integritas dan akurasi data

6. Produk perangkat lunak, artinya produk perangkat lunak yang diuji telah menjalani semua aktivitas QA perangkat lunak yang direncanakan, dengan semua cacat ditemukan tetap dan semua item dilaporkan sebagai ketidaksesuaian ditutup

7. Semua mesin klien, yang berada di tempat dengan konfigurasi yang ditentukan dan terhubung ke server melalui Internet atau intranet

Selain itu, pemeriksaan kesiapan pengujian sistem memastikan bahwa kondisi terpenuhi:



Gambar 4.7 Proses Inspeksi

1. Rencana uji dan kasus uji sudah ada, dan telah menjadi sasaran kegiatan QA seperti yang direncanakan. Semua cacat diperbaiki dan semua ketidaksesuaian ditutup.

2. Semua penguji telah diberi pengarahan tentang pengujian yang akan dilakukan.

3. Kriteria penutupan kegiatan pengujian disetujui dan diketahui oleh yang bersangkutan.

Inspeksi ini harus dilakukan oleh personel departemen QA. Sebagai alternatif, SPM atau PL dari proyek lain dalam organisasi dapat melakukan pemeriksaan ini. Laporan pemeriksaan kesiapan pengujian sistem disiapkan dan diserahkan kepada SPM atau PL proyek; laporan

tersebut mencantumkan cacat yang ditemukan, jika ada.SPM atau PL memperbaiki cacat dan menutup laporan inspeksi, untuk melanjutkan dengan melakukan pengujian sistem.

K. PEMERIKSAAN KESIAPAN PENGUJIAN PENERIMAAN

Pengujian penerimaan dilakukan sebagai prasyarat untuk mendapatkan penerimaan pelanggan atas pengiriman produk dan, dengan demikian, untuk menerima pembayaran. Ini adalah tahap yang sangat penting, karena setiap cacat yang ditemukan di sini mencerminkan kualitas organisasi yang buruk dan keterlambatan pengiriman, akibatnya menunda penerimaan pembayaran. Oleh karena itu, inspeksi yang dilakukan pada tahap ini memastikan bahwa semuanya benar dan siap untuk pengujian penerimaan.

Aspek-aspek berikut dipastikan selama inspeksi kesiapan pengujian penerimaan:

1. Perangkat lunak ini lengkap dalam segala hal.

2. Semua kegiatan QA yang direncanakan, termasuk inspeksi, tinjauan, dan pengujian lainnya, dilakukan secara menyeluruh, dan semua ketidaksesuaian yang dilaporkan telah diperbaiki dan ditutup dengan memuaskan.

3. Data uji telah dibuat dan diperiksa kesesuaiannya dan ditemukan akurat.

4. Rencana penerimaan yang disetujui sudah ada.

5. Kasus uji penerimaan yang disetujui siap digunakan.

6. Semua perangkat keras yang diperlukan, dengan konfigurasi yang tepat seperti yang ditentukan dalam rencana uji penerimaan, sudah ada.

7. Semua perangkat lunak sistem dan middleware serta database dimuat dengan benar pada perangkat keras dan berfungsi sebagaimana mestinya.

8. Data uji dimuat dan siap.

9. Konfirmasi pelanggan untuk melakukan pengujian penerimaan pada tanggal yang dijadwalkan diterima.

10. Semua entitas organisasi terkait diberitahu tentang jadwal pengujian penerimaan, dan semua dukungan yang diperlukan tersedia.

11. Rencana penggantian tersedia untuk memperbaiki bug yang ditemukan selama pengujian penerimaan serta untuk mengatasi kegagalan perangkat keras apa pun.

Pemeriksaan kesiapan pengujian penerimaan biasanya dilakukan oleh departemen QA. Dengan tidak adanya departemen QA, rekan SPM atau PL dari proyek lain dapat melakukan pemeriksaan ini. Laporan pemeriksaan kesiapan pengujian penerimaan disiapkan oleh pemeriksa dan diserahkan kepada SPM atau PL proyek; laporan tersebut mencantumkan cacat yang

ditemukan, jika ada. SPM atau PL mengatur perbaikan dari setiap cacat yang ditemukan dan menutup laporan inspeksi. Kemudian dilakukan pengujian penerimaan sesuai dengan yang direncanakan.

L. INSPEKSI KESIAPAN PENGIRIMAN

Tujuan pemeriksaan kesiapan pengiriman adalah untuk memastikan bahwa usia paket pengiriman memiliki semua komponen dan versi yang benar disertakan di dalamnya. Pengiriman untuk sebuah proyek dapat dilakukan dalam salah satu dari tiga mode:

(1) pengiriman single-shot, artinya semua komponen dikirimkan pada satu waktu;

(2) pengiriman sementara, artinya salah satu dari beberapa pengiriman

(3) pengiriman akhir, yang berarti pengiriman terakhir dari serangkaian pengiriman.

Aspek-aspek berikut tercakup dalam pemeriksaan kesiapan pengiriman:

1. Register konfigurasi proyek digunakan sebagai referensi.

2. Kode yang dapat dieksekusi, jika merupakan bagian dari pengiriman, diperiksa untuk tanggal dan waktu pembuatan dan untuk memastikan bahwa kode tersebut sesuai dengan register konfigurasi. Jika kebetulan ada "rutin pembangunan" yang digunakan untuk menyiapkan bangunan yang dapat dieksekusi, itu diperiksa untuk memastikan bahwa versi kode sumber dan pustaka yang tepat digunakan dalam mempersiapkan

3. Dalam pengiriman sementara, inspeksi memastikan bahwa semua komponen yang disebutkan dalam catatan pengiriman perangkat lunak sebenarnya termasuk dalam pengiriman mengatur.

4. Semua aktivitas QA yang direncanakan dilakukan pada semua komponen set pengiriman, dan semua cacat yang ditemukan ditutup.

5. Nomor versi masing-masing komponen set pengiriman diperiksa terhadap catatan pengiriman perangkat lunak dan register konfigurasi.

6. Dalam pengiriman akhir, inspeksi memastikan bahwa semua komponen yang disebutkan dalam pesanan pembelian pelanggan telah disertakan dalam pengiriman sebelumnya atau disertakan dalam pengiriman saat ini dan tidak ada komponen yang tersisa menunggu pengiriman ke pelanggan setelah pengiriman saat ini. Hal ini selain untuk memastikan aspek-aspek yang telah disebutkan sebelumnya.

7. Dalam pengiriman sekali pakai, pesanan pembelian pelanggan dan register konfigurasi digunakan sebagai referensi. Inspeksi memastikan bahwa semua komponen yang disebutkan dalam pesanan pelanggan termasuk dalam set pengiriman saat ini dan tidak diperlukan pengiriman lebih lanjut.Ini selain untuk memastikan nomor versi yang benar dan kebenaran rutin build, jika ada.

8. Media pengiriman adalah sebagaimana ditentukan dalam pesanan pembelian pelanggan, dan jumlah salinan yang tepat disertakan dalam set pengiriman.

9. Semua persetujuan yang diperlukan untuk melakukan pengiriman sudah tersedia.

10. Pengiriman dilakukan kepada orang yang tepat, sebagaimana ditentukan dalam pesanan pembelian pelanggan.

Pemeriksaan kesiapan pengiriman biasanya dilakukan oleh departemen QA. Terkadang personel departemen pemasaran juga mengambil bagian dalam jenis inspeksi ini. Dengan tidak adanya departemen QA, rekan SPM atau PL dapat melakukan pemeriksaan ini. Setelah pemeriksaan kesiapan pengiriman selesai, laporan pemeriksaan kesiapan pengiriman disiapkan dan diserahkan kepada SPM atau PL proyek; laporan mencantumkan cacat, jika ada. SPM atau PL mengatur perbaikan dari setiap cacat dan menutup laporan.

Pengiriman kemudian dilakukan ke pelanggan.Inspeksi di titik lain dalam fase pengembangan perangkat lunak dapat dilakukan sesuai kebutuhan tergantung pada sifat proyek dan kebutuhan inspeksi. Inspeksi ini ditentukan oleh SPM selama tahap perencanaan proyek, dan rinciannya dicatat dalam rencana QA perangkat lunak proyek.

M. PRAKTIK TERBAIK DALAM INSPEKSI

Sebagian besar organisasi perangkat lunak menggunakan inspeksi, karena inspeksi merupakan alat yang efektif untuk memastikan bahwa aktivitas utama dilakukan dengan lancar. Beberapa organisasi berpendapat bahwa inspeksi berlebihan, karena tujuan yang sama dicapai melalui audit akhir fase. Ini agak benar, tetapi audit terutama merupakan sistem verifikasi dokumen. Inspeksi tidak hanya melihat dokumen, tetapi juga memeriksa entitas fisik untuk memastikan kebenaran dan kesiapannya. Misalnya, inspeksi kesiapan pengujian sistem melihat register konfigurasi, register kerja, laporan tinjauan, dan log pengujian, untuk memastikan bahwa semua komponen telah dibuat dan semua aktivitas QA telah dilakukan secara lengkap. Itu juga melihat sistem untuk memastikan bahwa semua perangkat lunak sistem yang diperlukan dimuat dan bahwa semua data master sudah siap, bersama dengan status produk perangkat lunak saat ini pada sistem. Fungsi-fungsi ini digabungkan membuat inspeksi sangat diperlukan. Sebuah perangkap umum di banyak organisasi adalah pengecualian inspeksi sama sekali, dengan menggunakan argumen bahwa audit akhir fase mencapai tujuan yang sama.

Praktik terbaik untuk inspeksi jenis apa pun meliputi:

1. Melakukan inspeksi setidaknya sebelum pengujian sistem dan pengujian penerimaan dan sebelum mengirimkan produk perangkat lunak ke pelanggan.

2. Staf departemen QA dengan inspektur spesialis untuk melakukan inspeksi. Ini lebih efektif daripada meminta pengembang perangkat lunak dari dalam proyek atau dari proyek lain melakukan inspeksi.

Meminta pengembang perangkat lunak melakukan inspeksi seringkali berakhir dengan latihan mengisi formulir daripada inspeksi serius yang ditujukan untuk mengungkap kekurangan.

N. AUDIT

Audit digunakan terutama dalam organisasi yang memiliki proses pengembangan perangkat lunak yang telah diimplementasikan dalam proyek mereka. Audit adalah sistem verifikasi dokumen di mana dokumen dan catatan proyek dibandingkan dengan standar organisasi atau proses yang ditetapkan. Mereka umumnya berdurasi pendek, dengan sekitar satu hingga dua jam dihabiskan untuk mengaudit sebuah proyek atau suatu fungsi.

Audit digunakan sebagai alat QA terutama untuk memastikan kesesuaian pelaksanaan proyek dengan proses pengembangan perangkat lunak yang ditentukan organisasi. Mereka memastikan bahwa sebuah proyek sedang dieksekusi sesuai dengan proses yang ditetapkan organisasi dan siap untuk fase eksekusi berikutnya.

Audit dilakukan untuk tujuan mengungkap ketidaksesuaian (NCs), jika ada, dalam suatu proyek. Jika dokumen atau catatan proyek menunjukkan penyimpangan dari proses yang dijelaskan dalam proses yang ditetapkan organisasi, penyimpangan ini diperlakukan sebagai NC dan pelaksanaan proyek dianggap tidak sesuai dengan proses yang ditetapkan organisasi.

Keluaran dari suatu audit adalah laporan ketidaksesuaian (nonconformance report/NCR). NCR mencantumkan semua NC yang ditemukan selama audit. Gambar 4.8 menunjukkan contoh NCR.

Audit terdiri dari auditor (orang yang melakukan audit) dan auditee (orang yang proyeknya diaudit). Auditor harus memiliki pelatihan khusus dalam melakukan audit. Auditor yang melakukan audit dalam suatu organisasi harus menerima pelatihan audit internal, sedangkan auditor yang melakukan audit sertifikasi atau audit pengawasan untuk organisasi lain harus dilatih dan disertifikasi untuk melakukannya.

Durasi audit proyek yang biasa adalah satu hingga dua jam. Selama waktu ini, auditor memverifikasi semua dokumen proyek dan mencatat setiap NC yang ditemukan secara berurutan untuk menyiapkan NCR nanti. NCR yang sudah diisi diserahkan kepada auditee. Auditee kemudian harus mengambil tindakan yang diperlukan yang ditentukan dalam NCR untuk mengatasi NCs dan meminta mereka ditutup oleh auditor dalam waktu yang ditentukan.

Nonconformance Report

Project name:

Name(s) of the auditor(s):

Name of the auditee:

Date on which the audit is conducted: _____

From (time) _____ to (time) _____

Type of audit: ☐ Periodic/phase end ☐ Vertical/horizontal

Nonconformances uncovered during the audit (use an additional sheet if necessary)

NC no.	NC description	Reference to process for the NC	Closed on	Status (open or closed)

Opportunities for improvement observed, if any

Description of improvement opportunity	Artifact reference	Reference to section of the artifact for improvement opportunity

Gambar 4.8. Laporan ketidaksesuaian (halaman 1 dari 2)

Suggestions for improvement

Description of the suggestion	Reference to artifact or area for the suggested improvement

Signature of auditor: Date of signature:

Closure Action by the Auditee

Corrective actions implemented

Corrective action implemented	NC numbers covered by this corrective action	Comments

Preventive action implemented

Preventive action implemented	NC numbers covered by this corrective action	Comments

Signature of auditee: Date of signature:

Nonconformance closure actions (to be filled in by the auditor)

I have verified and found that all the nonconformances described above are closed satisfactorily, except the following, which are retracted/pending:

1.
2.
3.

Signature of auditor: Date of signature:

Gambar 4.9. Laporan ketidaksesuaian (halaman 2 dari 2)

Tindakan yang diperlukan sebagaimana ditentukan dalam NCR melibatkan hal berikut:

1. Melakukan tindakan korektif agar NC teratasi

2. Melakukan tindakan preventif agar NC tidak terulang di proyek nantinya

Gambar 4.10 menggambarkan proses audit.

Audit dapat diklasifikasikan dalam berbagai cara, seperti yang dibahas dalam bagian berikut.

O. AUDIT KESESUAIAN VERSUS AUDIT INVESTIGASI

Audit kesesuaian fokus pada kemanjuran implementasi proses organisasi selama pelaksanaan proyek. Mereka dilakukan untuk membandingkan dan membedakan dokumen proyek dengan proses organisasi, mengungkap NC, menyiapkan NCR, dan melacak NCR hingga resolusinya.

Audit investigasi biasanya berfokus pada menemukan penyebab kegagalan, tetapi terkadang fokus pada menemukan penyebab kesuksesan yang luar biasa. Dokumen pelaksanaan proyek diverifikasi dengan hati-hati dan wawancara mendalam diadakan dengan personel proyek untuk mengungkap alasan spesifik yang menyebabkan kegagalan atau kesuksesan besar. Mereka digunakan dalam skenario khusus saja.

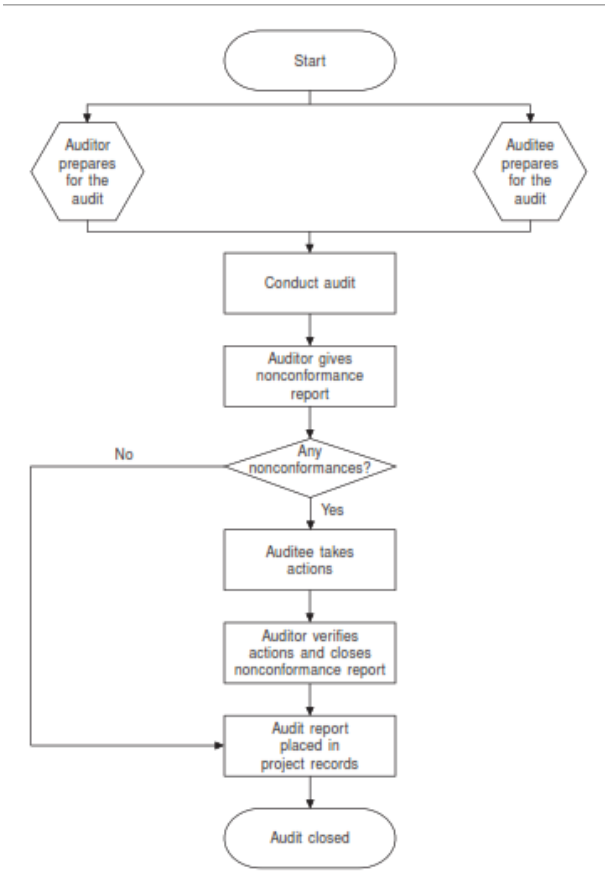
P. AUDIT VERTIKAL VERSUS AUDIT HORIZONTAL

Audit vertikal adalah audit kesesuaian yang dilakukan di seluruh organisasi pada beberapa proyek yang dipilih atau pada semua proyek; mereka fokus pada semua aspek proyek.

Audit horizontal juga merupakan audit kesesuaian yang dilakukan di seluruh organisasi pada beberapa proyek yang dipilih atau pada semua proyek, tetapi mereka hanya fokus pada satu aspek proyek. Audit manajemen konfigurasi adalah contoh yang baik dari jenis audit ini, yang dilakukan di sebagian besar organisasi. Audit horizontal fokus pada kemanjuran pelaksanaan salah satu aspek penting dari pelaksanaan proyek dalam organisasi.

Q. AUDIT BERKALA VERSUS AUDIT TAHAP-AKHIR

Audit berkala adalah audit kesesuaian yang dilakukan di tingkat organisasi berdasarkan durasi kalender. Biasanya, organisasi bersertifikasi ISO melakukan audit ini setiap dua atau tiga bulan kalender. Setiap audit mencakup beberapa proyek organisasi saat ini, dan semua proyek yang sedang dilaksanakan (bukan proyek tertutup) dalam organisasi dicakup dalam periode satu tahun. Pada akhir setiap siklus audit, temuan audit dikonsolidasikan dan disajikan kepada manajemen dan auditee.



Gambar 4.10 Proses Audit

Proses audit berkala adalah sebagai berikut:

1. Audit berkala biasanya dikoordinasikan di tingkat organisasi oleh departemen QA.

2. Setiap awal tahun, rencana audit disiapkan dan disetujui oleh otoritas yang berwenang. Rencana ini mencakup perincian seperti kemungkinan tanggal audit, kemungkinan proyek yang akan dimasukkan dalam setiap putaran audit, kemungkinan auditor, dll.

3. Pada setiap awal audit diadakan rapat pembukaan audit yang mengikutsertakan seluruh auditor, auditee, dan perwakilan manajemen. Perwakilan departemen QA menjelaskan proses audit secara umum, tujuan audit, proses resolusi NCR dan jadwal penutupan NC, dll. QA mengklarifikasi masalah yang diangkat oleh auditor atau auditee. Secara opsional, perwakilan manajemen juga menjelaskan sudut pandang manajemen tentang audit ini.

4. Auditor melakukan audit, sesuai dengan proses audit organisasi dan pedoman audit, dan mencatat NC yang ditemukan, jika ada. NCs dijelaskan kepada auditee dan NCR diserahkan kepada auditee. Departemen QA juga menerima salinan untuk tindak lanjut.

5. Departemen QA mengkonsolidasikan semua NCR dan melakukan analisis terhadap NC yang ditemukan dalam audit. Departemen QA melakukan pertemuan penutupan audit dengan peserta yang sama yang berada di pertemuan pembukaan audit dan mempresentasikan temuan audit serta analisis NCs. Kemanjuran implementasi proses dalam organisasi dibahas,

dan peluang untuk perbaikan dalam definisi proses atau implementasi disepakati. Departemen QA mencatat keputusan yang dibuat selama rapat, jika ada, dan melacaknya hingga resolusi.

6. Auditee menyelesaikan NCs dengan mengambil tindakan korektif dan juga melakukan tindakan pencegahan agar tidak terulang di proyek.

7. Auditee mendekati auditor untuk menutup NCR dan menunjukkan kepada auditor resolusi semua NC yang diajukan pada proyek.

8. Auditor memverifikasi keputusan, menutup NCR, dan menyerahkan NCR tertutup ke departemen QA.

9. Departemen QA mengkonsolidasikan semua NCR dan melakukan analisis resolusi NC. Analisis ini kemudian disajikan kepada manajemen pada kesempatan yang sesuai.

10. Audit berkala kemudian selesai.

Gambar 4.13 menunjukkan laporan audit konsolidasi yang akan disiapkan di akhir dari siklus audit berkala. Gambar 4.15 menggambarkan proses audit berkala.

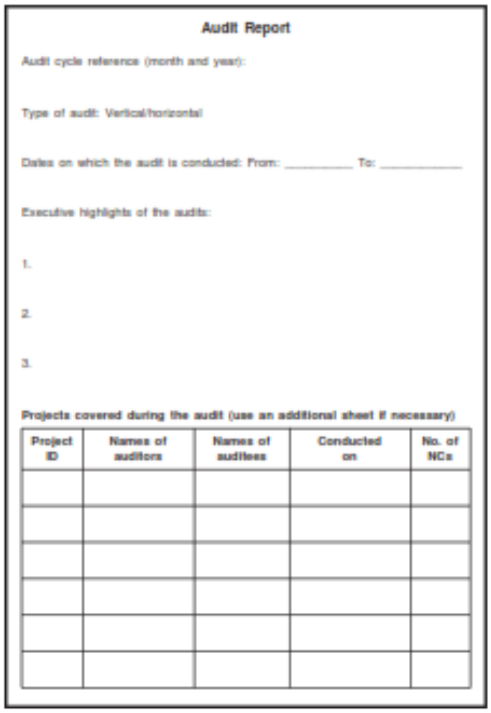
Audit akhir fase dipicu oleh peristiwa proyek. SPM mengatur ini audit berkoordinasi dengan departemen QA ketika fase pelaksanaan proyek selesai. Biasanya, audit ini dilakukan setelah fase berikut:

a. Inisiasi proyek— Audit ini dilakukan segera setelah proyek kegiatan inisiasi selesai. Ini memastikan bahwa proyek dimulai sesuai dengan proses organisasi untuk inisiasi proyek, yang memastikan bahwa fase berikutnya dari pengembangan perangkat lunak bergerak maju tanpa masalah. Audit inisiasi proyek dilakukan untuk semua proyek.
b. Analisis persyaratan perangkat lunak— Audit ini dilakukan pada proyek yang memiliki analisis persyaratan yang signifikan, tetapi akan dilewatiuntuk proyek yang lebih kecil atau jangka pendek. Ini memastikan bahwa prosesmengumpulkan dan menganalisis persyaratan proyek dilakukan di kesesuaian dengan proses organisasi untuk analisis kebutuhan perangkat lunak, bahwa aktivitas QA yang diperlukan telah dilakukan, dan bahwa NCs, jika ada, diselesaikan dengan benar. Analisis kebutuhan perangkat luna audit memastikan bahwa fase pengembangan perangkat lunak berikutnya bebas masalah.
c. Desain perangkat lunak— Audit ini dilakukan pada proyek-proyek yang memiliki aktivitas desain perangkat lunak yang signifikan, tetapi akan dilewati untuk proyek proyek berdurasi pendek atau proyek-proyek yang tidak memiliki pengaruh signifikan. komponen desain perangkat lunak. Audit memastikan bahwa desain perangkat lunak dilakukan sesuai dengan perangkat lunak organisasiproses desain, aktivitas QA yang diperlukan dilakukan, dan NCs, jika ada, diselesaikan dengan benar. Audit desain perangkat lunak memastikan: bahwa fase berikutnya tidak menghadapi masalah.
d. Konstruksi perangkat lunak— Audit ini dilakukan setelah perangkat lunak konstruksi selesai, yang berarti bahwa pengembangan perangkat lunak,tinjauan independen, dan pengujian unit dari kode yang dikembangkan selesai. Audit memastikan bahwa

konstruksi perangkat lunak dilakukan keluar sesuai dengan proses organisasi untuk perangkat lunak konstruksi dan integrasi itu, tinjauan sejawat, dan pengujian unit adalah dilakukan sesuai dengan proses organisasi untuk peer review dan pengujian unit. Audit konstruksi perangkat lunak dilakukan untuk semua proyek pengembangan perangkat lunak dan memastikan kesiapan produk perangkat lunak untuk pengujian sistem.

e. Pengujian sistem— Audit ini dilakukan setelah pengujian sistem selesai. Ini mungkin dilewati pada proyek-proyek yang kecil dan yang tidak memiliki komponen pengujian sistem yang signifikan, terutama ketika produk perangkat lunak diharapkan berfungsi pada satu platform saja. Audit memastikan bahwa pengujian sistem berhasil diselesaikan dan semua cacat yang ditemukan diselesaikan dengan memuaskan sesuai dengan proses pengujian sistem organisasi. Audit pengujian sistem memastikan kesiapan produk perangkat lunak untuk pengujian penerimaan oleh pelanggan.

f. Penutupan proyek— Audit ini dilakukan untuk semua proyek, tepat sebelum proyek ditutup secara resmi. Ini memastikan bahwa semua aktivitas penutupan proyek, seperti mendokumentasikan praktik terbaik dan terburuk proyek, mengarsipkan artefak, mengidentifikasi komponen yang dapat digunakan kembali dan menyerahkannya ke entitas organisasi yang sesuai, dan berbagi pengetahuan, dilakukan sesuai dengan proses organisasi untuk penutupan proyek. Audit penutupan proyek memastikan bahwa pengalaman proyek dibagikan dengan SPM lain dan didokumentasikan dengan baik untuk menjadi bagian dari gudang pengetahuan organisasi untuk referensi di masa mendatang.



Audit Report

Audit cycle reference (month and year):

Type of audit: Vertical/horizontal

Dates on which the audit is conducted: From: _____ To: _____

Executive highlights of the audits:

1.

2.

3.

Projects covered during the audit (use an additional sheet if necessary)

Project ID	Names of auditors	Names of auditees	Conducted on	No. of NCs

Gambar 4.13 Laporan audit konsilidasi (halaman 1 dari 2)

Types of nonconformances uncovered

Process area	No. of NCs	Comments*

* Describe if the nonconformances are due to drawbacks in the process, training, negligence, etc.

Proposed improvement actions

Proposed improvement action	Timeline

Any other significant points:

1.

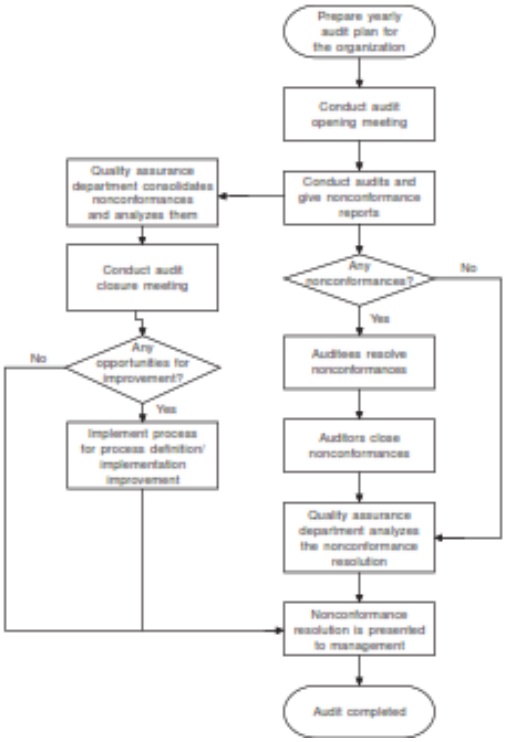
2.

3.

Signature of QA head:

Date of signature:

Gambar 4.14 Laporan audit konsilidasi (halaman 2 dari 2)



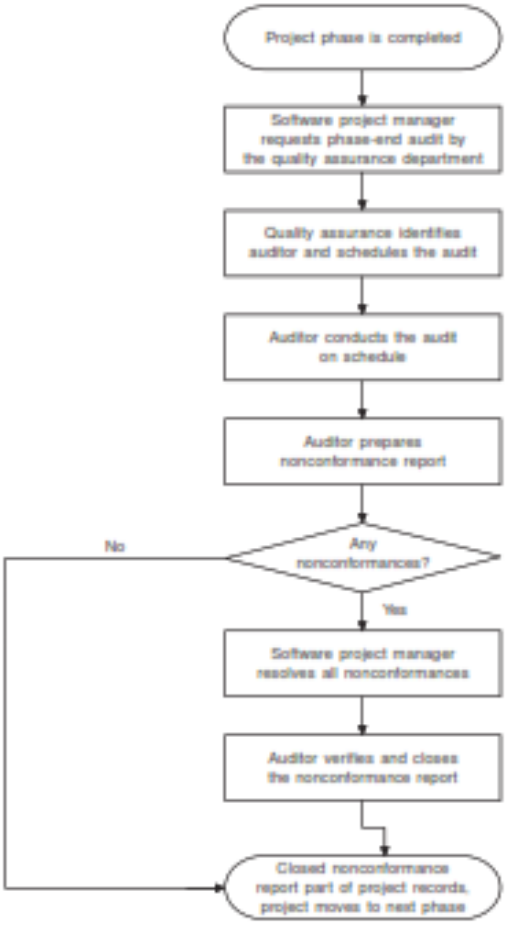
Gambar 4.15 Proses audit berkala

Langkah-langkah untuk melakukan audit akhir fase adalah sebagai berikut:

1. Ketika fase proyek selesai, SPM berkoordinasi dengan departemen QA untuk melakukan audit akhir fase yang sesuai.

2. Departemen QA mengidentifikasi seorang auditor, menugaskan audit kepada auditor yang dipilih, dan menjadwalkan audit dengan berkonsultasi dengan SPM.
3. Auditor melakukan audit sesuai jadwal, mencatat NCs, menyiapkan NCR, dan menyerahkan NCR ke SPM.
4. SPM menyelesaikan NCs dan, berkoordinasi dengan auditor, menutup NCR. Auditor memberikan izin untuk memindahkan proyek ke tahap berikutnya.
5. SPM menambahkan NCR tertutup ke catatan proyek, menyimpulkan: audit akhir fase.

Gambar 4.16 menggambarkan proses audit tahap akhir.



Gambar 4.16 menggambarkan proses audit tahap akhir.

R. AUDIT INTERNAL VERSUS AUDIT EKSTERNAL

Audit internal dapat berupa audit kesesuaian, audit investigasi, audit berkala, atau audit akhir fase, tetapi audit tersebut dilakukan oleh orang-orang internal organisasi. Namun, auditor internal independen dari proyek yang diaudit, artinya mereka dapat berasal dari departemen QA atau dari proyek lain. Audit internal dilakukan untuk

memastikan kesesuaian atau menyelidiki terjadinya peristiwa khusus. Proses audit mirip dengan proses yang dijelaskan sebelumnya.

Audit eksternal dilakukan oleh lembaga eksternal yang mengkhususkan diri dalam proses audit. Organisasi yang mencari sertifikasi untuk kepatuhan dengan seri standar ISO 9000 atau standar serupa lainnya menggunakan auditor eksternal Individu yang disertifikasi sebagai auditor utama dapat melakukan audit untuk sertifikasiatau untuk memastikan kepatuhan berkelanjutan dengan standar ISO. Secara opsional, sebuah organisasi dapat melibatkan konsultan eksternal untuk melakukan audit guna memperoleh opini yang tidak memihak tentang proses dan implementasinya dalam organisasi.

Terkadang audit eksternal dilakukan untuk memastikan kesiapan organisasi untuk audit sertifikasi. Audit eksternal terdiri dari proses yang sama dengan audit internal, dan NCR adalah sarana untuk mencatat dan melaporkan NCs yang ditemukan selama audit serta untuk menutup NCs.

Audit eksternal diklasifikasikan sebagai berikut:

a. Audit prasertifikasi —Audit ini dilakukan sebagai pendahuluan untuk audit sertifikasi. Ketika sebuah organisasi memiliki beberapa proyek yang berjalan secara bersamaan, konsultan eksternal dilibatkan untuk memastikan kesiapannya untuk audit sertifikasi. Audit prasertifikasi berfungsi sebagai latihan bagi organisasi agar dapat lulus audit sertifikasi. Hal ini juga memungkinkan organisasi untuk memuluskan setiap sisi kasar yang terbuka dalam persiapan untuk audit sertifikasi.
b. Audit sertifikasi dan sertifikasi ulang —Audit ini berujung padamemberikan atau menolak sertifikat yang didambakan kepada organisasi. Audit sertifikasi dilakukan hanya sekali, kecuali jika mengakibatkan penolakan sertifikat kepada organisasi. Sebagian besar model sertifikasi mutu mewajibkan sertifikasi ulang berkala, seperti setiap tiga tahun sekali. Audit sertifikasi dilakukan untuk organisasi yang belum pernah disertifikasi. Audit sertifikasi ulang sama persis dengan audit sertifikasi, kecuali dilakukan untuk organisasi yang telah disertifikasi.
c. Audit pengawasan—sertifikasi ISO 9000 mengamanatkan pengawasan audit setiap enam bulan sekali untuk memastikan implementasi proses itu berada pada level yang sama pada saat audit sertifikasi. Itu lembaga eksternal yang sama yang awalnya memberikan sertifikat melakukan audit pengawasan ini. Mereka sedikit diperkecil dari audit sertifikasi, dan sampel proyek yang lebih kecil diaudit.

Audit adalah alat yang sangat berguna untuk implementasi organisasi yang efektif proses, dan mereka digunakan di sebagian besar organisasi pengembangan perangkat lunak.

S. PRAKTIK TERBAIK DALAM AUDIT

Di banyak organisasi yang melakukan audit berkala, latihan menjadi lebih merupakan ritual daripada upaya sungguh-sungguh untuk menilai tingkat dan keseriusan implementasi proses dan kesesuaian di seluruh proyek dalam organisasi.

Setiap auditor biasanya mendaftar dua atau tiga NCs sehingga audit tampak otentik, tetapi sebagian besar NCs ditemukan secara informal ditunjukkan kepada

auditee dan tidak dicatat. Ketika semua NC tidak dicatat, analisis pasca audit tidak mencerminkan gambaran sebenarnya dari implementasi dan kepatuhan proses. Ini adalah perangkap paling umum yang saya amati dalam organisasi yang melakukan audit berkala.

Masalah ini dapat diatasi dengan beberapa cara. Pertama, termasuk dalam tim audit auditor spesialis yang bukan pengembang perangkat lunak. Auditor ini dapat berasal dari departemen QA atau dari luar organisasi. Dengan menggunakan temuan auditor spesialis ini, auditor sejawat dapat dinasihati tentang perlunya mencatat semua NC.

Cara lain untuk mengatasi masalah ini adalah dengan secara acak memasukkan NCR yang diserahkan oleh auditor rekan ke audit verifikasi oleh auditor spesialis. Fakta bahwa NCR mungkin dipilih untuk verifikasi acak akan memaksa auditor sejawat untuk rajin mencatat semua NC. Solusi kedua ini adalah praktik terbaik.

Jebakan lainnya adalah bahwa orang-orang yang bertanggung jawab untuk mengawasi pelaksanaan proses menganggap audit ini sebagai gangguan yang harus ditanggung daripada alat untuk menilai implementasi dan kepatuhan proses. Dalam satu perusahaan bersertifikasi ISO 9000, kepala teknis (yang biasanya bertanggung jawab atas pengiriman perangkat lunak ke pelanggan dan yang mengawasi semua pengembang perangkat lunak dalam organisasi) bahkan belum membaca proses organisasi! Jika orang yang paling senior tidak percaya pada implementasi dan kepatuhan proses, hasilnya hanya bisa berupa audit yang dilakukan sebagai ritual untuk tujuan pembuatan catatan.

Praktik terbaik adalah menempatkan orang-orang senior yang percaya pada kualitas dan implementasi proses dan kepatuhan. Jika mereka yang berada di posisi senior tidak percaya pada pengembangan perangkat lunak yang digerakkan oleh proses, mungkin lebih baik tidak melakukan audit sama sekali.

Jebakan umum dalam audit akhir fase adalah menugaskannya ke rekan dari dalam proyek itu sendiri. Ini juga menyebabkan audit memburuk menjadi latihan pembuatan catatan. Praktik terbaik adalah melibatkan auditor spesialis untuk melakukan audit akhir fase.

Dua argumen yang mendukung penggunaan auditor rekan adalah

(1) auditor memiliki kesempatan untuk mempelajari praktik terbaik dan perangkap dari proyek lain dan

(2) rekan sejawat lebih memahami implementasi proses daripada auditor spesialis. Kedua argumen tersebut valid jika dan hanya jika auditor sejawat menjalankan profesionalisme lengkap dan tidak menggunakan pendekatan quid pro quo . Rekan belajar dari proyek lain selama pertemuan penutupan audit, di mana temuan audit

dibagikan dengan semua orang yang terkait. Praktik terbaik adalah memiliki setidaknya beberapa auditor spesialis dalam tim audit.

Namun perangkap umum lainnya adalah organisasi menyalahkan SPM jika proyek menerima sejumlah besar NCs atau SPM memiliki kesan bahwa manajemen menyalahkan dia atas NCs. Konseling SPM yang proyeknya ditemukan memiliki sejumlah besar NC harus ditangani dengan sangat hati-hati. Manajemen senior perlu memastikan apakah jumlah NC terlalu besar sejak awal, karena jumlah absolut tidak menjelaskan keseluruhan cerita. Mengharapkan jumlah NC untuk proyek besar setara dengan proyek kecil tidak realistis. Manajemen senior juga perlu memastikan apakah NCs adalah kesalahan kebetulan dan kelalaian atau jika ada kelalaian. Konseling perlu diberikan hanya dalam kasus kelalaian, karena kesalahan kebetulan yang ditemukan oleh audit dikoreksi secara otomatis — lagipula, tujuan audit adalah untuk mengungkap kesalahan dan kelalaian yang tidak disengaja tersebut.

T. PROSES VERIFIKASI

Verifikasi adalah aktivitas QA yang sangat penting dalam organisasi pengembangan perangkat lunak, dan oleh karena itu tidak boleh dilakukan dengan pendekatan informal atau ad hoc. Organisasi pengembangan perangkat lunak profesional harus memiliki proses yang terdefinisi dengan baik untuk mendorong aktivitas verifikasi dalam organisasi. Proses terpisah diperlukan untuk setiap aktivitas verifikasi berikut:

1. Panduan
2. Inspeksi perangkat lunak
3. Audit

U. PROSES PANDUAN

Proses penelusuran menjelaskan jenis penelusuran yang dilakukan organisasi, peran dan tanggung jawab dalam mengatur dan melaksanakan penelusuran, dan analisis yang akan dilakukan terhadap cacat yang ditemukan dalam penelusuran.

Proses penelusuran juga mencakup prosedur untuk setiap jenis penelusuran:

1. Panduan mandiri
2. Panduan terpandu
3. Panduan grup

    Tinjauan pos

    Tinjauan rapat

Tinjauan rapat terpandu

4. Tinjauan ahli

Proses penelusuran juga mencakup pedoman untuk memilih jenis tinjauan yang akan dilakukan untuk artefak, peninjau yang direkomendasikan untuk setiap metode penelusuran, pedoman pertemuan, pedoman pelaporan cacat, dll. Selain itu, proses penelusuran menawarkan berbagai daftar periksa untuk setiap metode penelusuran. , serta format untuk laporan tinjauan, format susunan cacat untuk tinjauan grup, dan format lain yang spesifik untuk organisasi.

V. PROSES INSPEKSI PERANGKAT LUNAK

Proses inspeksi perangkat lunak menjelaskan jenis inspeksi perangkat lunak yang akan dilakukan dalam organisasi, peran dan tanggung jawab dalam mengatur dan melakukan inspeksi, pedoman pelaporan cacat, analisis yang akan dilakukan pada cacat yang ditemukan selama inspeksi, dll. Proses pemeriksaan perangkat lunak juga menawarkan prosedur untuk melakukan setiap jenis pemeriksaan perangkat lunak berikut ini:

1. Pemeriksaan kesiapan pengujian sistem
2. Pemeriksaan kesiapan pengujian penerimaan
3. Pemeriksaan kesiapan pengiriman
4. Pemeriksaan khusus organisasi lainnya

Ini juga menunjukkan format laporan inspeksi dan pedoman untuk melaporkan cacat yang ditemukan selama inspeksi.

W. PROSES AUDIT

Proses audit menjelaskan jenis audit yang akan dilakukan dalam organisasi serta peran dan tanggung jawab dalam meminta dan melakukan audit. Ini menawarkan prosedur untuk melakukan masing-masing jenis audit berikut:

1. Audit berkala
2. Audit fase-akhir
3. Audit investigasi
4. Audit organisasi lainnya

Proses audit juga menawarkan pedoman dan daftar periksa untuk melakukan setiap dari audit akhir fase berikut:

1. Inisiasi proyek

2. Analisis kebutuhan perangkat lunak

3. Desain perangkat lunak

4. Konstruksi perangkat lunak

5. Penutupan proyek

6. Audit akhir fase khusus organisasi lainnya

Selain itu, proses audit mencakup garis besar kursus pelatihan dan prosedur pelatihan untuk menginstruksikan auditor internal. Ini juga mencakup pedoman untuk berbagai audit yang akan dilakukan dalam organisasi, serta format NCR dan laporan audit yang akan digunakan untuk audit berkala. Contoh proses audit dan pedoman audit diberikan dalam Lampiran A.

X. PENERAPAN KEGIATAN VERIFIKASI DALAM PROYEK

Meskipun dimungkinkan untuk menerapkan kegiatan verifikasi dalam proyek menggunakan pendekatan ad hoc—dan beberapa organisasi melakukannya, hal itu tidak disarankan. Saat menggunakan pendekatan ad hoc, kegiatan verifikasi tidak direncanakan secara eksplisit dan SPM akan melaksanakan kegiatan verifikasi yang sesuai dengan urgensi pelaksanaan proyek.

Pendekatan yang lebih baik adalah pendekatan yang direncanakan. Saat menggunakan pendekatan ini, kegiatan verifikasi untuk suatu proyek direncanakan oleh SPM selama tahap perencanaan proyek, dan dicatat dalam rencana QA perangkat lunak. Rencana QA perangkat lunak, juga disebut sebagai rencana verifikasi dan validasi perangkat lunak. Ini mencatat rincian kegiatan verifikasi berikut yang akan dilaksanakan dalam proyek yang sedang direncanakan:

1. Daftar semua kegiatan verifikasi yang direncanakan untuk dilaksanakan diproyek

2. Daftar fase pelaksanaan proyek, setelah itu audit akhir fase akan dilakukan

3. Daftar tahapan di mana inspeksi perangkat lunak akan dilakukan

4. Artefak yang akan menjadi sasaran peer review, bersama dengan kemungkinan individu yang akan melakukan review dan jenis peer review yang akan digunakan

5. Metrik dan pengukuran yang harus dilakukan untuk memastikan: efektivitas kegiatan verifikasi

Kegiatan verifikasi yang direncanakan untuk suatu proyek harus sesuai dengan proses verifikasi organisasi. Tinjauan sejawat dari rencana QA perangkat lunak memastikan bahwa kegiatan verifikasi yang direncanakan benar-benar sesuai dengan proses verifikasi organisasi. Semua aktivitas ini harus diimplementasikan selama pelaksanaan proyek sesuai dengan rencana QA perangkat lunak. Kegiatan verifikasi juga harus mematuhi berbagai prosedur, pedoman, format, dan template yang ditetapkan untuk setiap kegiatan verifikasi yang direncanakan dalam proses verifikasi organisasi. Kemajuan dan status pelaksanaan kegiatan verifikasi dalam proyek harus dipantau bersama dengan kegiatan proyek lainnya, dengan menggunakan laporan status proyek sebagai mekanisme. Kegiatan ini tunduk pada pertemuan pemantauan kemajuan yang biasa diadakan oleh manajemen senior.

Selain kegiatan verifikasi yang direncanakan dalam rencana QA perangkat lunak untuk suatu proyek, kegiatan verifikasi lain yang direncanakan di tingkat organisasi juga harus dilakukan untuk proyek tersebut. Kegiatan verifikasi tersebut termasuk audit berkala, audit eksternal untuk sertifikasi atau pengawasan, audit horizontal seperti audit manajemen konfigurasi, dan lain-lain. Kegiatan verifikasi ini juga dapat mengungkap NC, dan SPM harus mengatur tindakan korektif dan pencegahan yang diperlukan untuk menutup NC.

Y. RINGKASAN

Verifikasi adalah kegiatan yang dilakukan untuk memastikan bahwa sesuatu sesuai dengan spesifikasi yang terdokumentasi, standar, peraturan, dan lain-lain. Tujuan verifikasi terutama untuk memastikan bahwa hal yang benar adalah dibangun, dan konfirmasi spesifik berikut dicari dari verifikasi: pencapaian fungsionalitas inti yang ditetapkan untuk artefak, Kelengkapan dan kelengkapan artifak, kesesuaian dengan standar dan pedoman yang ditetapkan untuk jenis artefak dalam rencana proyek, efisiensi dan efektivitas solusi yang disajikan oleh arti fakta, Kejelasan dan kebenaran artifak, Pencapaian semua fungsi tambahan yang diharapkan dari artefak, Peninjau mengonfirmasi bahwa semua fungsi tambahan terpasang dengan benar ke dalam artefak, tidak ada fungsi yang tidak perlu dalam artefak, tidak ada fungsionalitas berbahaya yang terkandung dalam artefak, format artefak mematuhi standar organisasi atau spesifikasi pelanggan, pemenuhan persyaratan fungsi hilir artefak ketika akan digunakan di hilir.

Z. SOAL LATIHAN

1) Apa yang dimaksud dengan verifikasi?Dan apa tujuan dari adanya verifikasi?
2) Apa yang dimaksud dengan walkthrough?Dan sebutkan jenis dari walkthrough!
3) Apakah perbedaan dari Audit Kesesuaian versus Audit Investigasi?
4) Sebutkan Kegiatan Verifikasi Dalam Proyek!
5) Sebutkan 4 proses audit!

BAB V
VALIDASI

A. DEFINISI VALIDASI

Validasi menunjukkan konfirmasi atau pembuktian suatu klaim. Dalam konteks pengembangan perangkat lunak, validasi mengacu pada aktivitas yang dilakukan pada produk perangkat lunak untuk mengkonfirmasi bahwa semua fungsi yang dirancang (atau diperlukan) memang dibangun dan bekerja sesuai dengan spesifikasi asli (penggunaan yang dimaksudkan), bersama dengan fungsi implisit lainnya. untuk memastikan keselamatan, keamanan, dan kegunaan.

Standar 610 dari Institute of Electrical and Electronics Engineers glosarium standar terminologi rekayasa perangkat lunak mendefinisikan istilah validasi sebagai "proses mengevaluasi sistem atau komponen selama atau pada akhir proses pengembangan untuk menentukan apakah memenuhi persyaratan yang ditentukan."

Dokumen model Capability Maturity Model Integration (CMMI®) untuk pengembangan (versi 1.2, Agustus 2006) mendefinisikan validasi sebagai “konfirmasi bahwa produk, sebagaimana disediakan (atau sebagaimana akan disediakan), akan memenuhi tujuan penggunaannya. Dengan kata lain, validasi memastikan bahwa 'Anda membangun hal yang benar.'” Ini juga menyatakan bahwa “tujuan validasi adalah untuk menunjukkan bahwa produk atau komponen produk memenuhi tujuan penggunaannya ketika ditempatkan di lingkungan yang dimaksudkan.”

Sinonim untuk kata memvalidasi antara lain mengotentikasi, mengesahkan, menguatkan, mengkonfirmasi, mendukung, menanggung, membuktikan, dan mendukung, antara lain. Salah satu definisi validasi adalah tindakan memastikan bahwa sesuatu itu valid.

Untuk memahami istilah “validasi” dengan benar, perhatikan skenario berikut yang melibatkan validasi:

a. Anda sedang membuat kontrak dengan seseorang. Sebelum Anda menandatangani kontrak, Anda menunjukkannya kepada pengacara untuk validasi (bukan verifikasi).Hanya setelah pengacara menyatakan bahwa itu sah (yaitu, sah secara hukum dan akan sah di pengadilan) barulah Anda menandatanganinya.

b. Anda menegaskan sesuatu secara tertulis. Anda memiliki notaris yang mengautentikasi afirmasi Anda. Setelah dikonfirmasi, penegasan Anda menjadi surat pernyataan dan sah secara hukum.

c. Anda sedang melakukan perjalanan ke Kutub Utara. Setelah mempelajari literatur tentang subjek, Anda membuat rencana. Anda membawa rencana Anda ke ahli di Kutub Utara dan meminta ahli untuk "mengkonfirmasi" bahwa rencana Anda masuk akal. Setelah ahli mengonfirmasi bahwa rencana Anda bisa diterapkan, Anda memulai petualangan Anda.

d. Anda telah menulis novel mata-mata-thriller. Sebelum Anda menerbitkannya, Anda memiliki mata-mata kehidupan nyata yang membacanya. Setelah mata-mata menyetujuinya, Anda melanjutkan dengan menerbitkan buku.

e. Sebuah pabrik mobil mengklaim bahwa mobil model barunya mampu mencapai tingkat konsumsi bahan bakar 230 mil pergalon. Departemen pemasaran menunjukkan klaim ini di depan sebuah pilih set jurnalis, dan jurnalis membuktikan dia.

Oleh karena itu, menurut definisi ini, validasi adalah tindakan pencegahan yang biasanya diambil sebelum mengambil risiko—terutama risiko strategis. Skenario di atas memiliki karakteristik berikut:

Ada klaim yang perlu validasi. Pemrakarsa klaim mengatur validasi. Validasi tidak dilakukan terhadap spesifikasi pencetus klaim. Itu dibuat berdasarkan spesifikasi yang mungkin telah dirumuskan oleh agen eksternal:

a. Dalam skenario pertama, kontrak disahkan melawan hukum tanah.

b. Dalam skenario kedua, penegasan divalidasi oleh individu independen yang diberi wewenang oleh pemerintah.

c. Dalam skenario ketiga, rencana divalidasi terhadap pengalaman langsung ahli.

d. Dalam skenario keempat, novel diautentikasi berdasarkan pengalaman langsung mata-mata itu.

e. Dalam skenario kelima, klaim disertifikasi melalui pengamatan langsung jurnalis.

Ada risiko yang memerlukan tindakan validasi.

Validasi memberi orang luar kepercayaan diri untuk dapat mengatakan "ini memang benar" atau "itu benar-benar berhasil." Validasi yang dilakukan dalam skenario pengembangan perangkat lunak melakukan fungsi yang sama yang tercantum di atas. Tim atau organisasi pengembangan perangkat lunak membuat klaim bahwa produk perangkat lunaknya berfungsi tanpa cacat. Klaim itu kemudian dibuktikan dengan validasi produk perangkat lunak.

Untuk mencapai nilai validasi penuh, tiga faktor berikut harus diterapkan:

1. Validasi dilakukan oleh orang independen yang bukan orang yang sama dengan yang mengajukan klaim.

2. Validasi dilakukan tidak hanya terhadap spesifikasi semut klaim, tetapi juga terhadap spesifikasi eksternal.

3. Validasi adalah upaya terencana dan terkoordinasi yang dilakukan dengan tujuan untuk membuktikan klaim dan menanamkan kepercayaan pada pemangku kepentingan; itu tidak dilakukan untuk keyakinan diri.

Selama pengembangan perangkat lunak, validasi artefak perangkat lunak penting dilakukan. Biasanya, desain perangkat lunak dan produk perangkat lunak divalidasi dalam skenario pengembangan kontrak. Dalam skenario produk komersial, spesifikasi produk divalidasi selain desain perangkat lunak dan produk perangkat lunak itu sendiri.

B. VALIDASI DESAIN PERANGKAT LUNAK

Mirip dengan pepatah “bukti puding ada di makan”, bukti bahwa desainnya kuat ada pada pembuatan produk dan kemudian pengujian produk. Bagaimana industri lain memvalidasi desain mereka? Dalam kasus di mana desain digunakan untuk menghasilkan produk dalam jumlah besar, prototipe (satu unit sampel produk) dibuat dan menjalani semua pengujian yang diperlukan, dan desain ditingkatkan berdasarkan hasil pengujian tersebut.

Metode ini sering digunakan dalam industri mobil dan elektronik, antara lain. Tapi bagaimana dengan industri seperti pembuatan kapal dan pembuatan pesawat? Mereka tidak mampu untuk membangun produk yang salah, bahkan untuk pengujian. Dalam industri seperti itu, mereka membuat prototipe model skala dan menguji model tersebut.

Sebelum membangun kapal besar, misalnya, sebuah perusahaan pembuat kapal membuat model kapal yang jauh lebih kecil yang rencananya akan dibangun dan menguji prototipe dalam lingkungan yang diperkecil. Hasil pengujian adalah untuk memvalidasi desain kapal yang diusulkan. Sebelum membuat model baru pesawat besar, pabrikan pesawat membuat prototipe yang lebih kecil dan lebih kecil dan mengujinya di terowongan angin. Berdasarkan hasil tersebut, desain diperbaiki dan produk dibangun.

Saat ini, model komputer memungkinkan pengujian desain melalui simulasi komputer, dan peningkatan desain dibuat berdasarkan simulasi tersebut. Namun vendor perangkat lunak yang menyediakan perangkat lunak simulasi untuk menguji roket, kapal, dan pesawat, misalnya, tidak menyediakan perangkat lunak simulasi yang memvalidasi desain perangkat lunak!Metodologi pengembangan yang menggunakan prototyping juga ada dalam pengembangan perangkat lunak. Metodologi ini menggunakan dua jenis prototipe: prototipe build-and-improve dan prototipe use-and-discard. Dalam prototipe build-and-improve, kerangka produk perangkat lunak, seperti tata letak layar, tata letak laporan, dan navigasi sederhana, dibangun di atas platform pengembangan yang sebenarnya. Pengembang perangkat lunak terus menyempurnakan desain berdasarkan hasil validasi prototipe. Dalam prototipe use-and-discard, prototipe dibangun di atas mockup menggunakan alat drafting.

Persyaratan divalidasi, dan kemudian produk perangkat lunak yang sebenarnya dibangun di atas platform pengembangan yang sebenarnya.Meskipun industri perangkat lunak saat ini menggunakan metodologi prototyping untuk memperoleh kebutuhan pengguna, tidak menggunakan satu untuk memvalidasi desain.

Desain perangkat lunak divalidasi melalui tinjauan kelompok oleh rekan atau pakar. Hanya jika anggota tim peninjau dipilih dengan cermat barulah metode ini menjadi efektif untuk memvalidasi desain perangkat lunak. Masuk akal jika tim peninjau terdiri dari lebih banyak pakar eksternal (di luar tim proyek), karena mereka akan memvalidasi produk tidak hanya terhadap standar internal tetapi juga terhadap persyaratan lapangan dan penggunaan lapangan. Masuk akal juga bagi pakar domain yang adalah pengguna produk perangkat lunak untuk dimasukkan sebagai anggota tim peninjau, bukan tim yang hanya terdiri dari perancang perangkat lunak.

Hasil validasi akan jauh lebih mendalam jika kedua metode pembuatan prototipe yang disebutkan di sini dan tinjauan kelompok menggunakan pakar domain digunakan untuk memvalidasi desain perangkat lunak. Faktanya, ini adalah praktik terbaik.

C. VALIDASI SPESIFIKASI PRODUK

Spesifikasi produk, terutama untuk produk perangkat lunak yang dibangun untuk memenuhi permintaan lebih dari satu pelanggan, sangat penting, karena semua aktivitas pengembangan perangkat lunak lainnya mengarah ke spesifikasi produk. Definisi spesifikasi yang tidak tepat memiliki dampak serius. Spesifikasi produk adalah pendahulu dari desain perangkat lunak di mana konstruksi produk bergantung.

Ketika spesifikasinya tidak tepat, jelas produk yang tepat tidak dapat dibangun. Karena mendefinisikan spesifikasi produk adalah langkah pertama dalam pengembangan perangkat lunak, spesifikasi produk perlu divalidasi.

Seperti halnya desain perangkat lunak, validasi spesifikasi produk juga sering dicapai melalui tinjauan kelompok yang menyertakan pakar domain. Tinjauan pos dan tinjauan rapat juga dapat digunakan. Keuntungan menggunakan tinjauan pos adalah bahwa pakar yang tersebar secara geografis dapat diikutsertakan, meskipun mungkin perlu waktu lebih lama untuk mendapatkan umpan balik dari mereka dan untuk menyelesaikan spesifikasi produk.

Keuntungan dari tinjauan rapat adalah finalisasi spesifikasi produk dapat dipersingkat, tetapi kelemahannya adalah bahwa semua ahli yang tersebar secara geografis harus diangkut ke organisasi dengan biaya organisasi. Alternatifnya adalah hanya menggunakan ahli lokal.Brainstorming adalah teknik lain yang digunakan untuk memvalidasi spesifikasi produk.

Dalam brainstorming, para ahli terkait berkumpul dalam pertemuan informal dan berunding untuk memvalidasi artefak yang ada. Brainstorming digunakan ketika kegiatan dilakukan untuk pertama kalinya dan dalam jenis kegiatan penelitian dan pengembangan.

D. VALIDASI PRODUK PERANGKAT LUNAK

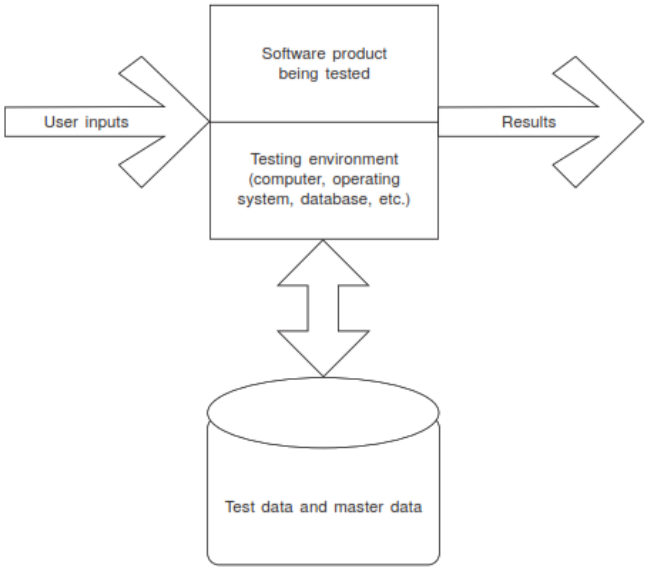
Pengujian perangkat lunak adalah alat utama untuk memvalidasi produk perangkat lunak akhir. Dalam Standar BS7925-1 dari British Standards Institution, pengujian didefinisikan sebagai "proses pelaksanaan perangkat lunak untuk memverifikasi bahwa itu memenuhi persyaratan yang ditentukan dan untuk mendeteksi kesalahan." Pengujian perangkat lunak didefinisikan sebagai "proses mengeksekusi item perangkat lunak untuk mendeteksi perbedaan, jika ada, antara perilakunya dan perilaku yang diinginkan."

Pengujian dilakukan dengan menggunakan satu set input yang dipilih yang hasil atau output yang diharapkan diketahui. Hal ini dilakukan terutama untuk menggali setiap dan semua cacat yang ada dalam sistem dan untuk mencegah produk cacat mencapai pelanggan.

Pengujian juga dilakukan untuk meyakinkan pelanggan bahwa produk sesuai dengan spesifikasi dan fungsionalitas yang telah disepakati sebelumnya. Pengujian digunakan untuk mengkonfirmasi kualitas daripada mencapainya. Itu dapat mendeteksi kesalahan, tetapi tidak dapat memastikan bahwa tidak ada cacat lain yang mengintai di produk perangkat lunak.

Pengujian menyeluruh berarti mencoba semua kemungkinan kombinasi input dan output, dan memastikan bahwa hasilnya benar. Gambar 5.1 menggambarkan pengujian perangkat lunak secara bergambar.

Pengembangan perangkat lunak adalah salah satu domain di mana fleksibilitas untuk pengujian tak tertandingi. Untuk menguji perilaku mobil dalam kecelakaan, mobil yang berjalan sempurna dihancurkan ketika menabrak sesuatu selama pengujian, yang mengakibatkan kerugian dari mobil; pengujian ulang mengharuskan mobil lain dibuat dan diuji. Menguji perilaku bangunan dalam tornado tidak mungkin karena tornado terbatas tidak dapat dibuat. Pengujian kemampuan suatu bangunan untuk menahan keadaan seperti itu harus dilakukan baik melalui simulasi komputer atau dengan model di laboratorium. Dalam domain perangkat lunak, bagaimanapun, perangkat lunak dapat dikenakan semua jenis pengujian tanpa takut merusak produk atau database atau melukai penguji. Fleksibilitas ini memungkinkan penggunaan pengujian perangkat lunak secara ekstensif sebagai alat utama untuk jaminan kualitas dalam domain pengembangan perangkat lunak.



Gambar 5.1 Pengujian perangkat lunak

Perangkat lunak juga kebetulan memiliki beragam fungsi, karena terkadang mencakup seluruh proses bisnis, seperti keuangan, manajemen material, atau manajemen pemasaran. Jumlah fungsi yang perlu diuji mungkin paling banyak dibandingkan domain lain, seperti manufaktur atau konstruksi. Fleksibilitas untuk pengujian lengkap ini digabungkan dengan berbagai fungsionalitas membuat pengujian menjadi aktivitas yang mahal, dan terkadang biaya pengujian sama atau bahkan melampaui biaya pengembangan itu sendiri.

Pengujian perangkat lunak diakui sebagai aktivitas penting dalam pengembangan perangkat lunak. Akhir-akhir ini, pentingnya pengujian independen (pengujian oleh orang yang tidak terlibat dalam pengembangan perangkat lunak) telah mendapatkan penerimaan yang lebih luas, sedemikian rupa sehingga perusahaan perangkat lunak yang mengkhususkan diri dalam pengujian perangkat lunak telah didirikan dan cukup sukses. Meningkatnya kompleksitas dan ukuran perangkat lunak telah menghasilkan pengujian yang lebih kompleks serta jenis pengujian yang lebih banyak.

E. MENGUJI BERBEDA JENIS PRODUK PERANGKAT LUNAK

Pengujian perangkat lunak tampak mudah, tetapi sebenarnya merupakan kegiatan yang kompleks karena mencakup beragam produk perangkat lunak, dan perangkat lunak yang perlu diuji beragam. Bagian berikut menjelaskan berbagai jenis perangkat lunak, yang semuanya memerlukan pengujian.

Sistem Pemrosesan Batch

Sistem perangkat lunak pemrosesan batch menerima input dari data yang disimpan dan memprosesnya dengan intervensi minimal dari pengguna untuk menghasilkan hasil. Contohnya termasuk sistem pemrosesan akhir pekan, sistem pemrosesan bulanan, sistem pemrosesan tahunan, dll. Sistem ini ditemukan dalam aplikasi biasa seperti fungsi cetak Microsoft Office

(mengonversi teks ke format yang dimengerti printer, mengirim umpan baris dan umpan halaman, memeriksa status printer dan pemadaman kertas, dll.) dan dalam aplikasi canggih seperti pemrosesan pembayaran di Internet (otorisasi, pendebitan akun, pengkreditan akun, transfer dana, dll.), penagihan elektronik (otorisasi, mengomunikasikan rincian tagihan, menerima pengakuan, menerima pembayaran, dll.), mengirim dan menerima email, dll.

Atribut utama dari sistem pemrosesan batch adalah bahwa data dimasukkan secara offline. Ketika data dikirimkan ke perangkat lunak, diharapkan telah divalidasi, disertifikasi untuk menjadi akurat, dan dalam format yang benar. Oleh karena itu, sistem pemrosesan batch tidak menyediakan validasi data yang ekstensif dan rutinitas penanganan kesalahan. Jika terjadi kesalahan, sistem pemrosesan batch gagal. Oleh karena itu, tindakan pencegahan utama yang perlu diambil dalam pengujian sistem pemrosesan batch adalah persiapan data yang akurat dalam format yang tepat yang diharapkan oleh perangkat lunak.

Pengujian black box (dijelaskan nanti dalam bab ini) adalah teknik pengujian utama untuk jenis perangkat lunak ini. Untuk melakukan pengujian struktural menggunakan teknik pengujian kotak hitam, data uji yang memaksa perangkat lunak untuk melintasi setiap jalur dalam kode harus dibuat. Seperti yang dinyatakan sebelumnya, aspek kunci dalam menguji sistem ini adalah menyiapkan data uji yang "benar" untuk melakukan pengujian yang menyeluruh dan untuk mengungkap semua kemungkinan kesalahan. Langkah-langkah pengujian identik dalam semua kasus uji; satu-satunya pembeda antara kasus uji adalah data uji. Persiapan data pengujian adalah tugas utama dalam pengujian sistem pemrosesan batch.

Dalam jenis perangkat lunak ini, pengujian unit adalah jenis pengujian yang biasanya dilakukan.Sistem pemrosesan batch dapat diintegrasikan ke dalam produk perangkat lunak lain, dan dalam kasus seperti itu, pengujian integrasi dilakukan pada produk perangkat lunak. Namun, satu set program pemrosesan batch dapat dijalankan dalam aliran, satu demi satu.

Setiap program diuji dalam pengujian unit, dan rangkaian diuji dalam "pengujian aliran", yang berarti semua program dijalankan dalam aliran. Hasil akhir diverifikasi untuk setiap perbedaan dari hasil yang diinginkan atau diharapkan.

Sistem Online

Sistem online juga disebut sebagai sistem yang dipicu peristiwa. Semua aplikasi online, di mana pengguna berinteraksi langsung dengan komputer melalui layar antarmuka pengguna grafis (GUI) untuk memasukkan atau mengambil data, adalah sistem online. Pengguna dapat mengikuti urutan apa pun untuk memasukkan data. Ketika pengguna mengklik antarmuka mouse untuk mengatur fokus ke kontrol di layar, sebuah peristiwa dipicu dan respons dihasilkan oleh perangkat lunak.

Contoh sistem jenis ini antara lain sistem reservasi online, sistem pembelian online, dan semua aplikasi bisnis online.Sistem online diharapkan penuh dengan validasi data dan rutinitas penanganan kesalahan, terutama dalam input. Untuk menguji sistem online, kasus uji adalah siap yang mengirim input positif yang membuktikan bahwa perangkat lunak melakukan apa yang seharusnya dilakukan dan yang mengirim input negatif yang menunjukkan bahwa

perangkat lunak memiliki semua validasi data dan rutinitas penanganan kesalahan untuk mencegah data yang salah masuk atau gagal sistem. Jenis sistem ini memerlukan pengujian ekstensif, termasuk semua jenis pengujian yang disebutkan selanjutnya dalam bab ini.

Dalam menguji sistem online, aspek kuncinya adalah desain kasus uji. Kasus uji banyak dan menghabiskan banyak waktu untuk merancang dan mendokumentasikan. Beberapa pengujian dilakukan secara intuitif menggunakan pedoman pengujian seperti pedoman pengujian GUI, pedoman pengujian negatif, dan pedoman pengujian laporan. Tes ini dijelaskan kemudian dalam bab ini.

Sistem Waktu Nyata

Sistem waktu nyata berinteraksi dengan mesin dan mengontrolnya untuk melakukan fungsi yang diinginkan. Mereka digunakan untuk aplikasi kontrol proses dalam industri manufaktur aliran proses, pesawat terbang, sistem senjata, dan mesin yang dikendalikan numerik komputer (mesin CNC).

Karena sistem waktu nyata berinteraksi dengan perangkat keras, terkadang tidak mungkin menggunakan perangkat keras dalam pengujian karena sinyal yang salah dari perangkat lunak dapat merusak mesin atau sekitarnya jika mesin tidak berfungsi. Oleh karena itu, respons mesin harus disimulasikan selama pengujian, dan test bed yang mensimulasikan respons dari mesin harus dibuat. Aspek penting di sini adalah pembuatan test bed, yang terkadang merupakan produk perangkat lunak yang menerima interupsi perangkat keras yang ditempatkan oleh perangkat lunak utama dan merespons interupsi tersebut persis seperti yang dilakukan mesin. Dengan demikian, untuk menguji perangkat lunak, perlu dikembangkan produk perangkat lunak lain.

Aplikasi Ilmiah

Sistem aplikasi ilmiah dibangun di sekitar persamaan matematika kompleks yang memproses algoritma besar dengan beberapa input. Mereka memproses intensif daripada data atau transaksi intensif. Mereka ditemukan dalam aplikasi seperti peramalan cuaca dan pemrosesan gambar.

Pekerjaan utama dalam menguji sistem ini adalah mempersiapkan hasil yang diharapkan dengan tangan. Untuk produk perangkat lunak yang mampu melakukan pemrograman linier dan aljabar matriks yang melibatkan seribu baris dan seribu kolom, mengerjakan solusi dengan tangan sangat membosankan, dan tanpa hasil yang diharapkan untuk dibandingkan, itu tidak akan pernah bisa diketahui. jika hasil yang sebenarnya akurat. Data dengan solusi yang diketahui dan telah dikerjakan sistem yang ada dapat diandalkan dapat digunakan dalam pengujian sistem perangkat lunak jenis ini. Untuk sistem berbasis komputer yang dikembangkan untuk pertama kalinya, sangat penting bahwa solusinya dikerjakan secara manual untuk memastikan bahwa hasil pengujian yang sebenarnya akurat. Presisi menjadi masalah saat menggunakan angka yang terdiri dari 16 angka penting atau lebih.

Aplikasi Seluler

Aplikasi seluler adalah sistem pemrosesan pesan yang digunakan terutama dalam komunikasi seluler. Perangkat lunak seluler harus samar dan singkat, karena sedikitnya jumlah memori yang tersedia di ponsel. Sekali lagi, untuk menguji perangkat lunak jenis ini, baik antarmuka ke perangkat keras diperlukan atau perangkat lunak uji yang mensimulasikan respons perangkat keras harus dikembangkan. Ketika pengujian sistem dilakukan, produk perangkat lunak yang dikembangkan harus dihubungkan dengan perangkat keras yang sebenarnya yang memfasilitasi komunikasi seluler.

Simulator Perangkat Lunak

Sistem ini adalah jenis lain dari sistem pemrosesan matematika yang, menggunakan perangkat keras khusus, mensimulasikan skenario kehidupan nyata. Simulator penerbangan adalah jenis perangkat lunak simulasi yang populer, tetapi ada banyak aplikasi praktis untuk simulasi di berbagai industri. Perangkat lunak simulator menawarkan grafis dan animasi.

Simulator juga berinteraksi dengan perangkat keras input dari sistem perangkat keras yang sebenarnya. Untuk menguji perangkat lunak ini, baik perangkat keras yang sebenarnya harus dibawa atau test bed yang menghasilkan sinyal yang identik dengan yang dihasilkan oleh perangkat keras yang sebenarnya harus disiapkan. Setelah test bed ini disiapkan, test case dapat dieksekusi, dan hasil yang diharapkan adalah hasil dari sistem yang sebenarnya. Aspek penting dalam pengujian perangkat lunak jenis ini adalah persiapan test bed.

F. PENGUJIAN DENGAN PERANGKAT KERAS KHUSUS

Pengujian jenis aplikasi yang dibahas di atas berbeda berdasarkan kelas mana dari aplikasi berikut ini:

(1) perangkat lunak yang dapat diuji tanpa perangkat keras khusus dan

(2) perangkat lunak yang memerlukan perangkat keras khusus untuk pengujian.

Aplikasi bisnis, baik sistem batch atau sistem online, dapat diuji tanpa perangkat keras khusus. Perangkat lunak waktu nyata, aplikasi ilmiah (yang terkadang tidak memerlukan perangkat keras khusus), aplikasi seluler, dan simulator semuanya memerlukan perangkat keras khusus. Perangkat keras khusus dalam konteks ini berarti perangkat keras selain sistem komputer biasa dan perangkat keras jaringan.

G. DASAR-DASAR PENGUJIAN

Ada enam prinsip yang berkaitan dengan pengujian perangkat lunak yang memandu organisasi:

1. Persyaratan pelanggan harus menjadi dasar untuk semua pengujian.

2. Pengujian perangkat lunak harus direncanakan sebelum pengujian dimulai.

3. Pengujian perangkat lunak tunduk pada prinsip Pareto atau "sedikit signifikan dan banyak tidak signifikan." Artinya, beberapa (sekitar 20%) unit atau modul mengandung sebagian besar (sekitar 80%) kesalahan.

4. Pengujian perangkat lunak harus dimulai dengan unit perangkat lunak terkecil dan berkembang secara bertahap ke seluruh sistem.

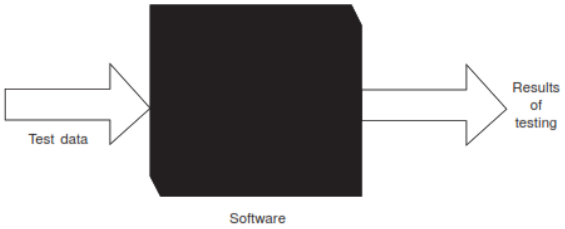
5. Pengujian menyeluruh (100%)—yaitu, menguji semua kasus yang mungkin—bukanlah praktis.

6. Agar efektif, pengujian perangkat lunak harus dilakukan oleh penguji independen yang tidak terlibat dalam pengembangan.

Ada dua jenis teknik pengujian: pengujian kotak hitam dan pengujian kotak putih, seperti yang dibahas pada bagian berikut.

H. PENGUJIAN KOTAK HITAM / BLACK BOX

Dalam pengujian kotak hitam, perangkat lunak diperlakukan sebagai "kotak hitam", dan logika internalnya untuk memproses data tidak dipertimbangkan. Satu set input diumpankan ke perangkat lunak, dan output yang dikirimkan oleh perangkat lunak dibandingkan dengan output yang diharapkan. Untuk menggunakan teknik ini, penguji mempertimbangkan fungsionalitas perangkat lunak dan mengelola pengujian. Pengujian black box digambarkan pada Gambar 5.2.



Gambar 5.2 Pengujian Black Box

Pengujian kotak hitam biasanya dilakukan dari antarmuka pengguna atau baris perintah. Program dipanggil, dan masukan yang diperlukan diberikan ke perangkat lunak sehingga ia memproses data uji dan masukan pengguna untuk menghasilkan keluaran yang dapat dibandingkan dengan keluaran yang diharapkan. Ini menentukan apakah perangkat lunak berfungsi dengan benar. Kemanjuran pengujian

kotak hitam tergantung pada perawatan dengan kasus uji dan data uji yang dirancang. Jika kasus uji yang lengkap, pengujian yang lengkap dan memiliki kesempatan yang lebih baik untuk mendeteksi

anomali dalam perangkat lunak. Desain kasus uji dibahas secara lebih rinci di bagian selanjutnya dari bab ini.

Berikut ini adalah langkah-langkah untuk melakukan pengujian black box:

1. Siapkan unit perangkat lunak untuk pengujian dengan membuat file yang dapat dieksekusi atau dengan menerima file yang dapat dieksekusi dari tim pengembangan, dan menginstalnya pada sistem pengujian.

2. Siapkan data master yang diperlukan untuk menjalankan pengujian. Data ini dapat disalin dari lingkungan pengembangan atau data master yang diperlukan dapat dimasukkan ke dalam file dan tabel data master.

3. Pelajari rencana pengujian dan catat tujuan pengujian.

4. Pelajari kasus uji yang dirancang untuk pengujian.

5. Jalankan program dari baris perintah atau antarmuka pengguna.

6. Jalankan kasus uji dalam urutan yang ditentukan. Di akhir setiap kasus pengujian, catat hasil aktual dan tentukan apakah eksekusi pengujian gagal atau lulus pengujian. Catat hasilnya.

7. Jika ragu dengan hasil eksekusi kasus uji, kembalikan data uji ke citra prates dan jalankan kembali pengujian.

8. Setelah semua kasus uji dieksekusi dan hasil aktual dicatat, bersama dengan keputusan lulus atau gagal, atur tinjauan manajerial dari hasil pengujian dan serahkan laporan kepada pembuat permintaan pengujian.

9. Memberikan klarifikasi kepada pencetus permintaan pengujian, dan membantu individu yang terlibat dalam penyelesaian cacat untuk memahami dengan benar cacat yang ditemukan selama pengujian.

10. Jika diperlukan, lakukan pengujian regresi untuk memastikan bahwa cacat diselesaikan dengan memuaskan. Ketika mereka telah diselesaikan, kosongkan unit perangkat lunak untuk tahap selanjutnya.

11. Jika pengujian regresi menemukan cacat baru atau cacat lama yang tidak teratasi dengan memuaskan, ulangi langkah 7 sampai 9 sampai semua cacat teratasi secara memuaskan.

Salah satu tindakan pencegahan yang harus diambil dalam pengujian kotak hitam adalah untuk

mempertahankan citra prates dari data master dan data uji, karena keduanya kemungkinan akan diubah. selama pengujian. Kapanpun pengujian ulang diperlukan, data master dan data pengujian harus disetel ke gambar pengujian awal.

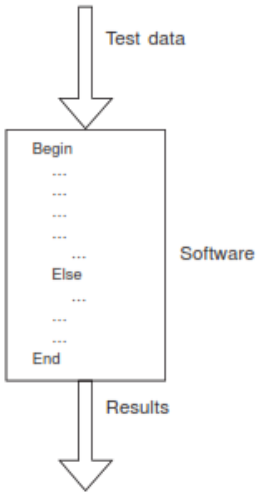
I. PENGUJIAN KOTAK PUTIH / WHITE BOX

Pengujian kotak putih mempertimbangkan logika internal dan pernyataan program perangkat lunak. Ini melibatkan melangkah melalui setiap baris kode dan setiap cabang dalam kode.

Untuk menggunakan teknik ini, penguji harus memiliki pengetahuan dalam bahasa pemrograman perangkat lunak dan harus memahami struktur program.

Pengujian kotak putih memastikan bahwa semua pernyataan program dan semua struktur kontrol diuji setidaknya sekali. Pengujian kotak putih, yang digambarkan pada Gambar 5.3, juga disebut sebagai pengujian kotak kaca.

Pengujian kotak putih dapat dilakukan dari baris perintah, pengguna di antarmuka, atau dari dalam program. Jika pengujian dilakukan dari baris perintah atau antarmuka pengguna, kelengkapan pengujian tergantung pada kasus uji untuk melintasi setiap jalur dalam perangkat lunak. Cara lainnya adalah dengan melakukan pengujian kotak putih menggunakan lingkungan pengembangan interaktif (IDE) atau debugger khusus bahasa. Alat-alat ini memiliki fasilitas untuk melakukan hal berikut:



Gambar 5.3 Pengujian White Box

a) Langkah melalui setiap baris kode

b) Tetapkan break point dalam kode tempat eksekusi menunggu penguji melanjutkan eksekusi

c) Tetapkan nilai awal atau ubah nilai variabel atau konstanta, tanpa perlu mengubah program

d) Lintasi setiap jalur untuk struktur kontrol dengan menetapkan nilai variabel kontrol secara dinamis

e) Hentikan eksekusi di titik mana pun dalam program dan lanjutkan pengujian dari awal atau di mana pun dalam program

f) Pindahkan eksekusi dari titik mana pun ke titik lain dalam program

Dengan menggunakan fasilitas ini, pengujian kotak putih mudah dilakukan. Bahkan, IDE atau debugger memungkinkan untuk menguji perangkat lunak secara menyeluruh pada tingkat pengujian

Saat pengujian kotak putih dilakukan dari IDE atau debugger, langkah-langkah berikut akan dilakukan:

1. Siapkan data master dan muat ke dalam file atau tabel data master.

2. Pelajari rencana pengujian dan catat tujuannya.

3. Pelajari pedoman pengujian yang berlaku, jika ada.

4. Dapatkan daftar periksa yang berlaku dan simpan agar tetap berguna.

5. Pelajari kasus uji dan bersiaplah untuk mengeksekusinya.

6. Menerima unit perangkat lunak yang akan diuji dari tim pengembangan, dan memuatnya ke sistem pengujian.

7. Luncurkan IDE atau debugger, dan buka unit perangkat lunak yang akan diujidi IDE atau debugger.

8. Tetapkan titik istirahat di mana jeda dalam eksekusi program diinginkan.

9. Jalankan kasus uji dalam urutan yang ditentukan dan catat yang sebenarnya hasil.

10. Tentukan hasil lulus atau gagal untuk setiap kasus uji dengan membandingkan hasil aktual dengan hasil yang diinginkan, dan mencatat hasil lulus atau gagal.

11. Jalankan semua pengujian seperti yang disarankan dalam pedoman pengujian yang berlaku dan daftar periksa, jika ada, dan catat hasilnya.

12. Mengatur tinjauan manajerial dari hasil pengujian, dan menyerahkan laporan pengujian kepada pembuat permintaan.

13. Bantu tim resolusi cacat dalam memahami cacat dengan benar laporan.

14. Saat diminta, lakukan pengujian regresi untuk memastikan kepuasan resolusi semua cacat.

15. Jika pengujian regresi menemukan cacat baru atau cacat asli yang tidak diselesaikan dengan

memuaskan, ulangi langkah 12 hingga 14.

Terkadang IDE atau debugger mungkin tidak tersedia untuk melakukan pengujian kotak putih. Dalam kasus seperti itu, perangkat lunak uji yang akan menguji unit perangkat lunak yang dipertimbangkan perlu dikembangkan.

Perangkat lunak pengujian tersebut melakukan fungsi-fungsi berikut:

Ini memanggil unit perangkat lunak yang sedang dipertimbangkan dan melewati parameter yang diperlukan untuk itu.Ini menerima nilai akhir yang dihasilkan oleh unit perangkat lunak yang dipertimbangkan dan menyajikannya kepada penguji untuk evaluasi kemanjurannya. Ini memungkinkan interaksi, jika diperlukan, dengan unit perangkat lunak yang dipertimbangkan selama eksekusi.

Ketika IDE atau debugger tidak dapat digunakan untuk menampilkan nilai antara atau untuk menangguhkan dan melanjutkan eksekusi, pernyataan sementara mungkin harus dimasukkan ke dalam unit perangkat lunak yang sedang dipertimbangkan. Dalam kasus seperti itu, penguji juga harus seorang pemrogram, dan pengujian harus dilakukan pada salinan program, bukan program aslinya.

J. PENDEKATAN UNTUK PENGUJIAN

Ada dua pendekatan dasar untuk pengujian perangkat lunak:

1) Pengujian intuitif

2) Pengujian berbasis proses

1) Pengujian Intuitif

Pengujian intuitif dilakukan oleh penguji berpengalaman yang menggunakan akal sehatnya. Deskripsi umum tentang fungsionalitas dan saran atau panduan untuk pengujian intuitif yang menjelaskan cara menggali cacat mungkin tersedia. Pengujian dilakukan dengan menggunakan pengalaman dan intuisi penguji. Sejumlah kreativitas atau akal sehat diharapkan dari penguji. Misalnya, saat menguji layar input, penguji memasukkan nilai berdasarkan label yang disediakan untuk kotak input. Mudah untuk menguraikan apa yang harus dimasukkan:

a) Jika label menunjukkan tanggal, masukkan tanggal yang sesuai dengan layar. Misalnya label seperti tanggal lahir, tanggal lamaran, tanggal nikah, dll mudah dikenali, sehingga memudahkan dalam menentukan input data yang benar dan salah.

b) Jika label menunjukkan data numerik seperti gaji, harga, biaya, panjang, tinggi, berat, kode pos, dll, mudah menentukan input data yang benar dan salah.

c) Jika label menunjukkan referensi ke data alfanumerik, seperti nama seseorang, alamat orang tersebut, gelar, ID email, kota, dll., sekali lagi, mudah untuk menentukan input data yang benar dan salah.

d) Jika label menunjukkan membuat pilihan menggunakan tombol radio atau kotak centang, mudah untuk menentukan input yang benar dan yang salah.

e) Penguji dapat memastikan bahwa ketika input data yang salah diberikan, perangkat lunak menolaknya.

f) Penguji memverifikasi apakah data yang dimasukkan disimpan dan diambil dengan benar.

g) Jika menguji laporan, penguji menghasilkan laporan yang berisi data yang diinginkan dan memastikan hal-hal berikut:

    a. Kolom disejajarkan dengan benar.

    b. Tidak ada data yang terpotong atau terpotong.

    c. Semua total akurat.

    d. Judul laporan, judul halaman, nomor halaman, dan tanggal laporan ditampilkan dengan benar dan lengkap, tanpa kesalahan ejaan.

h) Jika menguji layar pertanyaan, penguji menghasilkan beberapa pertanyaan dan memastikan bahwa informasi yang diperlukan diambil dan ditampilkan dengan benar.

i) Penguji, menggunakan akal sehat, menentukan kondisi kesalahan, menghasilkan log pengujian, memberikan log pengujian kepada pembuat artefak kode, dan mengatur penyelesaian cacat.

j) Keuntungan dari pendekatan pengujian ini adalah bahwa sebagian besar waktu yang biasanya dihabiskan untuk perencanaan pengujian dan desain kasus uji dapat dihemat. Kerugian dari pendekatan ini untuk pengujian termasuk yang berikut:

k) Ini membutuhkan penguji berpengalaman untuk melakukan tes dan mengungkap semua cacat yang mengintai.

l) Hampir tidak mungkin untuk menguji semua fungsi tanpa rencana pengujian.

m) Hampir tidak mungkin untuk melakukan pengujian menyeluruh, bahkan untuk fungsi-fungsi kritis.

Beberapa organisasi masih menggunakan pendekatan ini untuk pengujian. Saya telah melihatnya digunakan dalam sebuah organisasi yang melakukan validasi independen untuk produk yang diproduksi oleh perusahaan lain, dan organisasi tersebut mensertifikasi mereka dengan cukup sukses.

2) Pengujian Berbasis Proses

Dalam pendekatan ini, pengujian perangkat lunak dilakukan sesuai dengan proses validasi yang ditentukan menggunakan rencana pengujian dan serangkaian kasus uji. Untuk setiap proyek,

strategi pengujian perangkat lunak diputuskan selama tahap perencanaan proyek, dan dicatat dalam rencana pengujian atau rencana jaminan kualitas perangkat lunak.

Strategi pengujian dijelaskan di bawah ini.

a) Strategi Uji

Strategi pengujian berkaitan dengan mengungkap sebanyak mungkin cacat dalam waktu dan anggaran biaya yang dialokasikan dan dengan memaksimalkan dampak pengujian tersebut. Strategi pengujian untuk suatu proyek umumnya termasuk dalam rencana pengujian untuk proyek tersebut.

Langkah pertama dalam menyelesaikan strategi pengujian adalah menetapkan tujuan pengujian, seperti berikut ini:

1. Sasaran mutu—Ini berhubungan dengan tingkat cacat yang ditemukan: Mengungkap semua cacat, terlepas dari waktu atau biaya Mengungkap hampir semua kemungkinan cacat dalam waktu yang tersedia, dengan biaya dan waktu sebagai kriteria utama Mengungkap semua kemungkinan cacat dalam waktu yang tersedia, dengan waktu menjadi kriteria utama

2. Tujuan penerimaan pelanggan—Tujuan utama pengujian adalah untuk meyakinkan pelanggan bahwa produk dibuat sesuai dengan persyaratan pelanggan dan bahwa semua fungsi berfungsi tanpa cacat dan untuk mendapatkan persetujuan pelanggan dan dibayar oleh pelanggan untuk produk tersebut.

3. Tujuan sertifikasi produk— Pengujian dilakukan seperti yang ditentukan oleh pelanggan, dan produk disertifikasi sesuai permintaan pelanggan. Produk dapat disertifikasi di salah satu area berikut: Bebas virus dan spyware Fungsionalitas Kegunaan Perbandingan dan posisi relatif Peringkat produk.

Selain tujuan, berikut juga merupakan bagian dari strategi pengujian:

a. Jenis pengujian yang akan disertakan dalam pengujian—Menentukan pengujian mana yang perlu dilakukan pada produk perangkat lunak untuk mencapai tujuan kualitas proyek.

b. Cara menguji—Tentukan metodologi pengujian, seperti: Pengujian berbasis rencana dan uji kasus atau pengujian intuitif Pengujian kotak putih atau kotak hitam Pengujian manual atau pengujian berbasis alat

c. Pengujian regresi —Menentukan jumlah iterasi untuk pengujian regresi (hanya sekali atau diulang sampai semua cacat diperbaiki) .

d. Kriteria keberhasilan penyelesaian pengujian—Tentukan kriteria agar pengujian dinyatakan berhasil. Terkadang waktu yang berlalu adalah kriteria untuk penyelesaian pengujian. Artinya, pengujian sebanyak mungkin dilakukan hingga tenggat waktu tercapai. Kriteria lainnya adalah pengujian dilakukan sampai anggaran yang telah ditetapkan dihabiskan. Lain adalah pengujian dilakukan sampai tidak ada lagi cacat yang

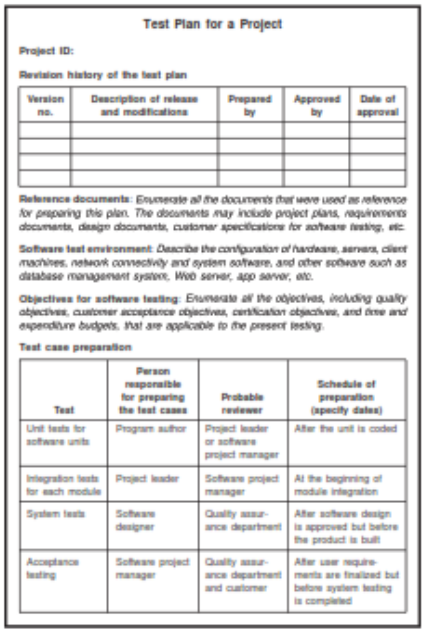
ditemukan. Yang lainnya adalah semua kasus uji yang ditentukan dieksekusi. Kriteria seperti itu akan ditentukan dan dicatat dalam rencana pengujian atau rencana jaminan kualitas perangkat lunak.

e. Mekanisme untuk penutupan dan eskalasi cacat, bila perlu— Mekanisme mencakup siapa yang akan menutup cacat, bagaimana menutup cacat, siapa yang akan mengeskalasi, kapan dan kepada siapa harus mengeskalasi, bagaimana mengeskalasi, dll.

f. Pelaporan kemajuan—Untuk dilakukan selama pelaksanaan proyek.

g. Analisis cacat —Tentukan apakah analisis seperti analisis ABC, analisis kategori, analisis kekritisan cacat, dll. diperlukan.

Strategi pengujian untuk suatu proyek didokumentasikan dalam rencana pengujian untuk proyek tersebut. Gambar 5.4 menunjukkan contoh format rencana pengujian.



Test Plan for a Project

Project ID:

Revision history of the test plan

Version no.	Description of release and modifications	Prepared by	Approved by	Date of approval

Reference documents: *Enumerate all the documents that were used as reference for preparing this plan. The documents may include project plans, requirements documents, design documents, customer specifications for software testing, etc.*

Software test environment: *Describe the configuration of hardware, servers, client machines, network connectivity and system software, and other software such as database management system, Web server, app server, etc.*

Objectives for software testing: *Enumerate all the objectives, including quality objectives, customer acceptance objectives, certification objectives, and time and expenditure budgets, that are applicable to the present testing.*

Test case preparation

Test	Person responsible for preparing the test cases	Probable reviewer	Schedule of preparation (specify dates)
Unit tests for software units	Program author	Project leader or software project manager	After the unit is coded
Integration tests for each module	Project leader	Software project manager	At the beginning of module integration
System tests	Software designer	Quality assurance department	After software design is approved but before the product is built
Acceptance testing	Software project manager	Quality assurance department and customer	After user requirements are finalized but before system testing is completed

Gambar 5.4 Format rencana pengujian untuk suatu proyek (halaman 1 dari 4)

Tests to be conducted for the project

Test	Probable testers	Test objectives	How to test	Criteria for completion
Unit testing	Independent peers from the project team	Ensure that the code is defect-free	Based on test cases and white box testing using IDE	All defects uncovered are closed
Integration testing	Project leader	Ensure that the interfaces are working without defects	Every time a unit is integrated with the module using black box testing and test cases	All defects uncovered are closed
System testing	Project leader and software project manager for the project	Ensure that the product works on Windows 2000, XP, and Vista, using Explorer, Firefox, and Chrome	Using system testing test cases	All defects uncovered are closed

Regression testing strategy: *Specify whether regression testing, whenever required, is to be conducted until all defects uncovered, whether in the main test or in the regression test, are closed or if regression testing is to be conducted only once after all defects are closed. If regression testing is conducted only once, specify the strategy to handle residual defects uncovered in regression testing.*

Escalation mechanisms: *Describe the criteria for escalation of unresolved issues in testing or defect closure. Include situations where the author disputes the defect, defect closure is not satisfactory, the defect classification is disputed, etc. Specify the executives, both in the delivery department and the quality assurance department, to whom issues can be escalated and the mechanism for communicating the escalation, such as e-mail, phone call, progress-monitoring meeting, etc. Also specify the timeline allowed for resolution of an escalation.*

Progress reporting: *Specify the timeline for reporting progress and status of testing (such as weekly or daily), to whom the progress report is to be communicated, responsibility for preparing the report, and so on.*

Gambar 5.4 Format rencana pengujian untuk suatu proyek (halaman 2 dari 4)

Risks identified: *Enumerate all the risks identified, along with their mitigation plan activities*

Risk ID	Risk description	Probability of occurrence	Mitigation plan

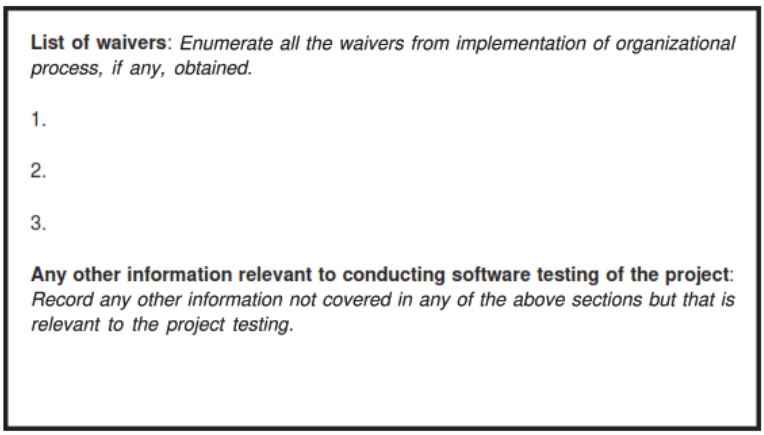
Tools to aid in testing

Name of tool	Purpose	Administrator	Reference to tool documentation
PMPal	Defect reporting, resolution, and defect metrics	Software project manager	Project information folder
IDE for programming languages	Unit testing	Program author(s)	Available with the IDE itself
Microsoft Office Suite	Prepare test cases, test logs, and reports and carry out analyses	Concerned persons	Available inside the suite itself
Doors	Load testing	Software project manager	Available in the tool itself

Proposed analyses for the project

Analysis	Person responsible for carrying out the analysis	Schedule for carrying out the analysis
Defect injection rate for programmers	Quality assurance department	Once every calendar month on the last working day
Rework effort for defect resolution	Quality assurance department	Once every calendar month on the last working day
Defect category analysis	Quality assurance department	Once every calendar month on the last working day
Defect by origin analysis	Project leader or software project manager	Once every calendar month on the last working day

Gambar 5.4 Format rencana pengujian untuk suatu proyek (halaman 3 dari 4)



List of waivers: *Enumerate all the waivers from implementation of organizational process, if any, obtained.*

1.

2.

3.

Any other information relevant to conducting software testing of the project: *Record any other information not covered in any of the above sections but that is relevant to the project testing.*

Gambar 5.4 Format rencana pengujian untuk suatu proyek (halaman 4 dari 4)

b) DESAIN KASUS UJI

Setelah strategi pengujian dan rencana proyek ditentukan dan dicatat dalam rencana jaminan kualitas perangkat lunak, untuk setiap pengujian yang dimaksudkan harus ada satu set kasus uji yang digunakan untuk pengujian.

Institute of Electrical and Electronics Engineers Standard 610 mendefinisikan test case sebagai "satu set input pengujian, kondisi eksekusi, dan hasil yang diharapkan yang dikembangkan untuk tujuan tertentu, seperti untuk menjalankan jalur program tertentu atau untuk memverifikasi kepatuhan dengan persyaratan tertentu, dan sebagai dokumentasi menentukan input, hasil prediksi, dan serangkaian kondisi eksekusi untuk pengujianbarang." Dari definisi ini, berikut ini dapat disimpulkan tentang kasus uji:

a) Kasus uji digunakan untuk menjalankan pengujian pada produk perangkat lunak.

b) Kasus uji terdiri dari input pengguna yang disediakan untuk aplikasi dan prosedur untuk mengeksekusi test case selama uji.

c) Kasus uji merinci keluaran yang diharapkan dari perangkat lunak saat: pengujian dijalankan dengan input pengguna yang ditentukan.

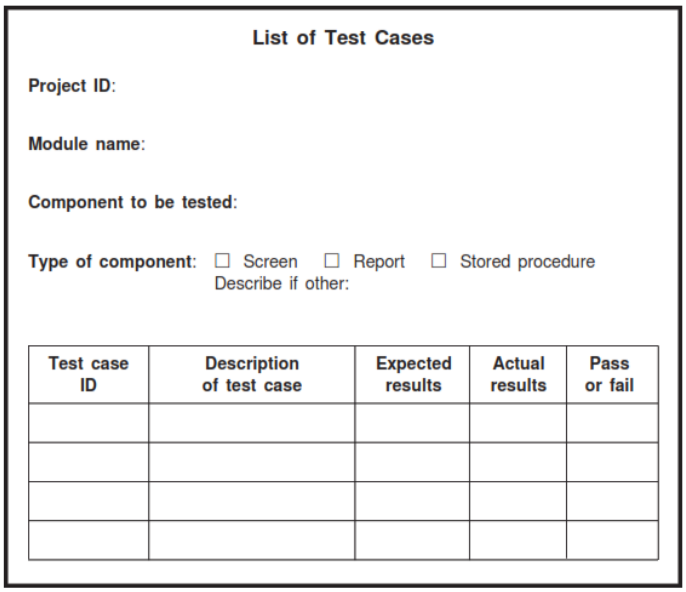
d) Kasus uji khusus untuk artefak kode perangkat lunak atau kelas perangkat lunak artefak kode.

e) Kasus uji memfasilitasi dan memastikan kepatuhan artefak kode perangkat lunak dengan persyaratan tertentu.

f) Kasus uji didokumentasikan.

Praktik yang biasa dilakukan adalah mendokumentasikan kasus uji dalam spreadsheet seperti:Microsoft Excel atau dalam alat seperti TestPal. (Alat ini tersedia gratisunduh dari Pusat Sumber Daya Unduhan Nilai Tambah Web diwww.jrosspub.com.) Praktik umum adalah

mendokumentasikan kasus uji untuk satu komponen perangkat lunak (software code artifact) dalam satu dokumen. Gambar 5.5 menunjukkan contoh format definisi kasus uji.



List of Test Cases

Project ID:

Module name:

Component to be tested:

Type of component: ☐ Screen ☐ Report ☐ Stored procedure
Describe if other:

Test case ID	Description of test case	Expected results	Actual results	Pass or fail

Gambar 5.5 Format definisi kasus uji

Tabel kondisi adalah cara lain untuk menggambarkan kasus uji. Sebuah tabel kondisimenggambarkan perilaku sistem untuk kombinasi input yang berbeda. Untukcontoh, di layar masuk, pengguna memasukkan ID pengguna dan kata sandi dan mengklik pada tombol OK atau tombol Batal. Ketika tombol Batal adalah diklik, tindakan masuk harus dibatalkan, tetapi ketika pengguna mengklik OK tombol, sistem berperilaku seperti yang dijelaskan pada Tabel 6.1.

Desain kasus uji sangat penting dalam pengujian perangkat lunak. Kasus uji yang dirancang dengan benar dapat mengungkap semua cacat, dan kasus uji yang dirancang dengan buruk akan hilang sisa cacat pada produk perangkat lunak. Tujuan dari desain kasus uji adalah untukmengungkap semua cacat yang mengintai di perangkat lunak dan untuk menguji seluruh perangkat lunak sepenuhnya, dengan kendala minimalisasi usaha dan waktu.

Kasus uji harus diturunkan dari artefak informasi perangkat lunak. Tabel 5.6 merupakan daftar yang membantu dalam menurunkan kasus uji.

Tabel 5.6 Tabel kondisi untuk layar masuk

Condition	User enters user ID and password and clicks OK button
Valid user ID and valid password	Accept
Valid user ID and invalid password	Reject and prompt for valid password
Invalid user ID and valid password	Reject and prompt for valid user ID
Invalid user ID and invalid password	Reject and prompt for valid user ID and password
Empty user ID and valid password	Reject and prompt for user ID
Valid user ID and empty password	Reject and prompt for password
Empty user ID and empty password	Reject and prompt for user ID and password

Jumlah kasus uji yang akan dirancang dan didokumentasikan cukup besar. Pertimbangkan implikasi berikut dalam merancang kasus uji: Untuk setiap input data numerik (termasuk data tipe tanggal), lima tes kasus, menggunakan teknik analisis partisi dan nilai batas dijelaskan nanti di bagian ini, diperlukan.

Tabel 5.7 Informasi perangkat lunak yang membantu dalam menurunkan kasus uji

Type of test	Information artifacts	Remarks
User acceptance testing	User requirements documents	User acceptance testing needs to prove that all user requirements are met by the software
System testing	Software design description—high-level design (software requirements specification)	Target system specification portion of design documents gives this specification
Integration testing	Software design description—high-level design (software requirements specification)	Interface description portion of design documents gives this specification
Unit testing	Software design description—low-level design or detailed design	Specification for the unit in the design gives this specification

Pemeriksaan ukuran harus dilakukan untuk semua data nonnumerik, satu per item data.

Semua data nonnumerik harus diperiksa untuk memastikan telah dimasukkandan bahwa area entri tidak boleh dikosongkan.

Pengujian logika diperlukan untuk memeriksa keberadaan data yang tidak valid, seperti dua titik desimal dalam data numerik, karakter numerik dan khusus dalam bidang data nama, dll.

Oleh karena itu, set kasus uji bahkan untuk unit yang cukup kompleks sangat besar. Proyek modern berukuran besar, dan upaya yang diperlukan untuk menyiapkan rangkaian kasus uji yang lengkap sangat ekstensif. Untuk alasan ini, biasanya untuk menyiapkan kasus uji di mana diharapkan penguji tidak dapat secara intuitif mengetahui kasus uji sendiri. Pengujian berbasis pedoman biasanya digunakan untuk jenis pengujian perangkat lunak berikut:

a) Pengujian GUI

b) Pengujian navigasi

c) Tes negative

d) Uji beban

e) Tes stress

f) Pengujian paralel dan pengujian bersama

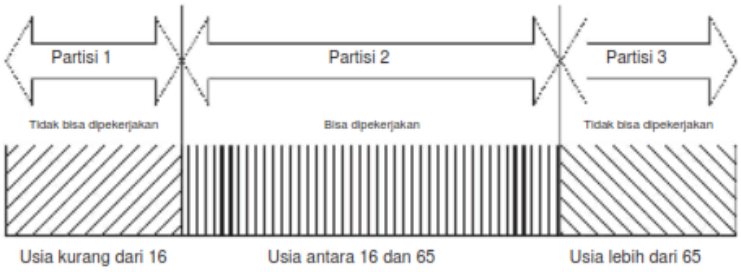
Organisasi menggunakan pedoman ini untuk menghindari keharusan menyiapkan kasus uji yang lengkap. Pengujian integrasi, pengujian sistem, dan pengujian penerimaan biasanya dilakukan terhadap kasus uji.

Beberapa teknik untuk desain kasus uji yang membantu memastikan pengujian itu:kasus yang komprehensif dibahas dalam bagian berikut.

K. PARTISI KESETARAAN

Dalam partisi ekivalensi, ruang input dipartisi menjadi input yang valid dan input yang tidak valid. Contoh berikut mengilustrasikan teknik ini. Dalam aplikasi sumber daya manusia, usia karyawan dapat minimal 16 (minimal usia kerja) dan maksimal 65 (usia pensiun). Partisi nilai yang valid adalah antara 16 dan 65. Ada dua partisi dengan nilai yang tidak valid: satu di bawah 16 dan yang lain di atas 65. Oleh karena itu, ada tiga partisi untuk kasus ini—satu valid dan dua tidak valid. Satu test case dapat dirancang untuk setiap partisi, menghasilkan tiga test case.

Hasil yang mungkin adalah sebagai berikut:



Gambar 5.8 Partisi kesataran

Jika kondisi input menentukan rentang nilai (seperti 16 hingga 65), ada tiga partisi, seperti yang dinyatakan di atas.

Jika kondisi input menentukan satu nilai (seperti 16), akan ada tiga partisi: satu partisi yang valid, satu partisi yang tidak valid di atas nilai yang valid, dan satu partisi yang tidak valid di bawah nilai yang valid.

Jika kondisi input menentukan nilai Boolean (benar atau salah), akan ada dua partisi: satu partisi yang valid dan satu partisi yang tidak valid.

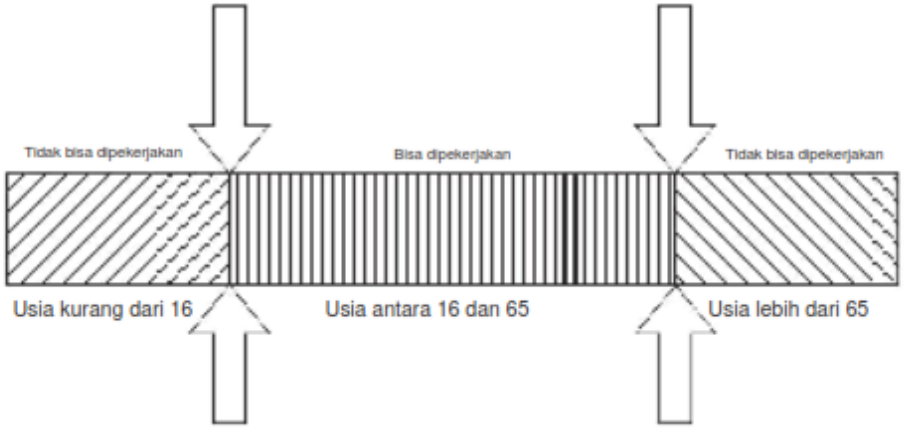
Jika kondisi input menentukan sekumpulan nilai yang valid, akan ada satu partisi untuk setiap nilai yang valid dan satu partisi yang tidak valid untuk nilai yang tidak valid.

L. ANALISIS NILAI BATAS

"Serangga mengintai di sudut dan berkumpul di perbatasan," pernyataan bijak yang dikaitkan dengan Boris Beizer, adalah dasar dari teknik ini. Sekali lagi menggunakan contoh pembagian kesetaraan, ada dua batasan: usia kerja minimum dan usia kerja maksimum (usia pensiun). Kedua batas ini—16 dan 65—harus diterima, dan semua nilai di bawah 16 dan di atas 65 harus ditolak. Oleh karena itu, kasus uji dirancang yang menggabungkan teknik partisi ekivalensi dan analisis nilai batas. Dalam contoh ini, ada lima kasus uji:

1. Satu nilai antara 16 dan 65—nilai valid

2. Satu nilai di batas bawah 16—nilai valid



Gambar 5.9 Nilai Batas

3. Satu nilai tepat di bawah batas bawah (yaitu, kurang dari 16)— nilai tidak valid (biasanya nilai ini diberikan sebagai 1 hari kurang dari 16)

4. Satu nilai pada batas atas 65—nilai valid 5. Satu nilai tepat di atas batas atas (yaitu, lebih besar dari 65)—nilai tidak valid (biasanya nilai ini diberikan sebagai 65 tahun dan 1 hari)

M. KESALAHAN MENEBAK

Seperti yang diterima secara umum bahwa pengujian menyeluruh dari semua skenario yang mungkin tidak praktis, organisasi mencoba memastikan produk bebas cacat dengan merancang kasus uji untuk contoh di mana cacat mungkin terjadi. Umumnya, pedoman untuk menebak kesalahan dikembangkan. Dalam teknik ini, perancang kasus uji menggunakan pengalaman, intuisi, dan akal sehat untuk merancang kasus uji yang mungkin mendeteksi kesalahan.

Seperti namanya sendiri, ada sejumlah tebakan yang terlibat, yang berarti insinyur perangkat lunak cenderung melakukan kesalahan. Menggunakan teknik ini tentu membutuhkan pengalaman bertahun-tahun dalam mengembangkan dan menguji perangkat lunak.

Contoh area di bidang tanggal di mana kesalahan mungkin terjadi termasuk tanggal yang tidak valid seperti 30 Februari yang dimasukkan, bulan yang disetel ke 13, atau tahun yang salah seperti 9999 yang dimasukkan. Contoh dalam aplikasi sumber daya manusia adalah memasuki usia yang tidak valid untuk bekerja. Kronologi adalah area lain di mana kesalahan mungkin muncul. Misalnya, ketika penerimaan material harus mendahului pengeluaran material dan tanggal pengeluaran sebelum tanggal penerimaan, transaksi harus ditolak oleh sistem. Contoh lain adalah angka negatif yang dimasukkan di mana hanya angka positif yang diharapkan.

Pengembang perangkat lunak merancang kasus uji untuk mendeteksi kesalahan dalam perangkat lunak menggunakan firasat dan intuisi yang sangat berkembang berdasarkan pengalaman pengembangan perangkat lunak selama bertahun-tahun. Organisasi mencatat semua area kesalahan yang mungkin terjadi dan menyiapkan pedoman tebakan kesalahan dan daftar periksa untuk digunakan dalam merancang kasus uji.

N. CAKUPAN LOGIKA

Dalam metode ini, logika desain perangkat lunak digunakan untuk menurunkan kasus uji yang mengevaluasi perangkat lunak untuk memastikan bahwa logika tersebut menghasilkan hasil yang diinginkan.

Kasus uji yang dirancang membuktikan atau menyangkal logika yang dibangun ke dalam desain perangkat lunak.

O. PEMERIKSAAN KONSISTENSI

Pemeriksaan konsistensi melibatkan perancangan kasus uji yang memeriksa konsistensi pemrosesan dari berbagai titik dalam perangkat lunak. Misalnya, dalam aplikasi perangkat lunak manajemen informasi gudang, saldo stok untuk suatu barang dapat diperoleh dari salah satu dari berikut ini:

1. Permintaan stok untuk suatu item
2. Laporan stok bulanan
3. Buku besar toko harga
4. Masalah material
5. Laporan kekurangan bahan

Meskipun masing-masing proses ini dapat diatur oleh unit perangkat lunak yang berbeda, masing-masing menyediakan informasi saldo stok, dan saldo stok untuk suatu item harus sama, terlepas dari proses dari mana ia diperoleh.

Konsistensi informasi dari unit-unit yang berbeda ini diperiksa. Jika ada perbedaan nilai yang dihasilkan, itu merupakan indikasi yang jelas dari kesalahan pada unit perangkat lunak tersebut.

Oleh karena itu, kasus uji dirancang untuk memastikan konsistensi nilai. Mereka berlaku di mana pun ada persyaratan untuk menampilkan informasi yang sama melalui fungsionalitas perangkat lunak yang berbeda.

P. PELACAKAN PERSYARATAN

Pelacakan persyaratan adalah metode yang paling umum untuk merancang kasus uji. Persyaratan, mulai dari spesifikasi kebutuhan pengguna hingga desain detail perangkat lunak, ditelusuri, dan kasus uji untuk menguji produk perangkat lunak dirancang untuk mengonfirmasi bahwa perangkat lunak memang memenuhi dan memenuhi persyaratan pengguna.

Contoh komponen layar masuk berikut mengilustrasikan bagaimana kasus uji diturunkan dari persyaratan perangkat lunak dengan menelusurinya melalui desain tingkat tinggi, desain tingkat rendah, dan akhirnya desain kasus uji. Tabel 5.3 menunjukkan bagaimana persyaratan ini berubah dalam berbagai hal.

Kasus uji diperlukan untuk menguji layar masuk untuk memastikan layar berfungsi untuk memenuhi kebutuhan pengguna dan seperti yang dirancang. Di sini kebutuhan pengguna dibagi menjadi tiga layar dalam desain tingkat tinggi:

Satu layar memastikan bahwa kata sandi yang dipilih kuat dan menangkap pertanyaan dan jawaban yang akan digunakan untuk mengambil kata sandi yang terlupakan.Layar kedua memastikan bahwa hanya pengguna yang berwenang yang mengakses aplikasi.

Tabel 5.1 Transformasi persyaratan untuk layar masuk

Perbedaan	Keterangan
Persyaratan pengguna	Akses ke fungsionalitas produk harus dibatasi oleh ID pengguna dan sistem kata sandi. Itu harus cukup aman untuk mencegah pengguna yang tidak sah mengakses sistem. Itu juga harus membatasi penyusup untuk meretas keamanan melalui entri coba-coba dari ID pengguna dan kombinasi kata sandi. Pada saat yang sama, pengguna yang berwenang harus dibantu dengan cara yang nyaman

	untukmengambil ID pengguna atau kata sandi yang terlupakan.
Desain tingkat tinggi (spesifikasi persyaratan perangkat lunak)	Definisi layar detail pengguna mencapai keamanan kata sandi dengan memastikan bahwa kata sandi adalah kombinasi huruf dan angka dan bukan kata kamus. Layar menangkap pertanyaan dan jawaban untuk mengambil ID pengguna dan kata sandi. Layar “Re trieve user ID/password” membantu pengguna mendapatkan kembali user ID atau password yang terlupakan. Layar masuk mencapai fungsionalitas mengizinkan atau membatasi akses ke pengguna. Layar ini memiliki fasilitas untuk memasukkan user ID dan password. Itu harus memiliki dua tombol: satu untuk mengirimkan entri ke sistem dan satu untuk membatalkan entri. Harus ada tautan untuk mengambil ID pengguna atau kata sandi. Hanya tiga upaya dari layar yang diizinkan untuk entri ID pengguna dan kata sandi yang tidak valid.
Deskripsi desain perangkat lunak (desain tingkat rendah atau desain detail) untuk komponen layar masuk	Tata letak layar (tidak disajikan di sini) Spesifikasi program: 1. Tampilan layar dan atur fokus ke kotak ID pengguna. 2. Atur jumlah percobaan ke nol. 3. Jika tombol OK diklik, maka lakukan langkah 4 sampai 10 di bawah ini. 4. Tingkatkan jumlah percobaan secara bertahap satu per satu. 5. Jika jumlah percobaan lebih besar dari tiga, maka tampilkan pesan “Mencapai jumlah percobaan maksimum. Silakan hubungi administrator Anda. ” Nonaktifkan kolom entri. 6. Verifikasi apakah ID pengguna kosong. Jika

	demikian, atur fokus ke kotak ID pengguna dan tampilkan pesan "ID Pengguna kosong." 7. Verifikasi apakah kata sandi kosong. Jika demikian, atur fokus ke kotak kata sandi dan tampilkan pesan "Kata sandi kosong." 8. Buka tabel pengguna dan verifikasi apakah ID pengguna ada di tabel. Jika ID pengguna tidak ditemukan, setel fokus ke kotak ID pengguna dan tampilkan pesan “ID pengguna salah. Silakan coba lagi.” 9. Jika ID pengguna ditemukan di tabel pengguna, verifikasi apakah kata sandi yang diberikan oleh pengguna sama dengan yang ada di tabel. Jika kata sandi tidak cocok, atur fokus ke kotak kata sandi dan tampilkan pesan “Kata sandi salah. Coba lagi." 10. Jika kata sandi yang diberikan oleh pengguna cocok dengan yang ada di tabel, tutup layar masuk dan tampilkan layar berikutnya. 11. Jika link “Lupa user ID atau password” diklik, tutup layar login dan tampilkan “Retrieve user ID or password” layar.

Q. PEMERIKSAAN WAKTU RESPONS

Waktu respons sangat penting dalam sistem perangkat lunak waktu nyata yang mengontrol mesin seperti pesawat, kapal, dan roket, serta robot dan mesin otomatis. Waktu respons juga menjadi sangat penting untuk berbasis Internet bisnis yang menawarkan fitur online seperti pemesanan reservasi, pembelian, pemrosesan pembayaran, dll. Misalnya, memasukkan nomor kartu kredit untuk melakukan pembayaran; waktu respons yang lama dapat menyebabkan pengguna bertanya-tanya:

Apakah pembayaran telah dilakukan dan kartu kredit saya ditagih?

Jika saya mengklik dua kali, apakah kartu kredit saya akan dikenakan biaya lagi untuk pembelian yang sama?

Apakah sistem hang?

Mungkinkah kartu kredit saya sudah ditagih tapi pembayaran belum diterima?

Apakah saya akan mendapatkan apa yang saya bayar?

Dalam operasi belanja online, jika sistem membutuhkan waktu lama untuk ditampilkan, pelanggan mungkin menutup situs Web dan pindah ke situs pedagang lain atau mungkin sama sekali meninggalkan gagasan belanja online. Oleh karena itu, selain sistem waktu nyata, waktu respons menjadi penting untuk sistem bisnis online.

Kasus uji dirancang untuk menguji waktu respons produk perangkat lunak untuk memastikannya dalam batas yang dapat diterima. Waktu respons dipahami sebagai waktu yang berlalu antara saat pelanggan memberikan perintah (mengklik tombol OK atau Kirim atau menekan tombol Enter) dan perangkat lunak mulai menampilkan respons. Aturan praktis dalam sistem bisnis online adalah bahwa waktu respons maksimum yang diizinkan harus antara 15 dan 30 detik. Waktu respons diperiksa dengan menggunakan stopwatch yang menunjukkan detik dalam seperseratus detik atau dengan menyisipkan pernyataan dalam program untuk menampilkan waktu dari segera setelah permintaan dikirim hingga segera setelah respons selesai. Waktu respons perangkat lunak diperoleh dengan mengurangkan waktu akhir dari waktu awal. Memulai dan menghentikan stopwatch harus dilakukan dengan cepat saat menggunakan metode ini untuk memeriksa waktu respons. Dalam metode tampilan sistem, perangkat lunak perlu diubah untuk menampilkan waktu jam sistem segera setelah perintah diberikan dan segera setelah perangkat lunak merespons, sehingga waktu yang dibutuhkan sistem untuk menghasilkan respons dapat diselesaikan.

R. LINGKUNGAN UJI

Lingkungan pengembangan perangkat lunak diatur selama tahap inisiasi proyek dan biasanya mencakup item berikut:

1. Server untuk database, server Web, dan tingkat menengah

2. Mesin klien, satu untuk setiap pengembang

3. Jaringan untuk semua mesin

4. Kit alat pengembangan perangkat lunak untuk mengembangkan perangkat lunak

Langkah-langkah berikut kemudian dilakukan:

1. Muat semua artefak informasi perangkat lunak, seperti dokumen persyaratan, dokumen desain, standar dan pedoman, dll., ke dalam area umum yang dapat diakses oleh semua anggota tim.

2. Mengatur prosedur manajemen konfigurasi untuk menangani kontrol perubahan dan promosi kode ke tahap berikutnya. Biasanya, kode akan berpindah dari status pengembangan ke status pengujian unit, lalu ke status integrasi produk, status pengujian integrasi, status pengujian sistem, status pengujian penerimaan, dan akhirnya ke status pengiriman. (Subjek manajemen konfigurasi berada di luar cakupan buku ini.)

3. Siapkan lingkungan pengujian yang diperlukan untuk melakukan serangkaian pengujian minimum, khususnya pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian penerimaan. Jika jenis pengujian lain direncanakan, lingkungan pengujian juga perlu disiapkan untuk pengujian tambahan tersebut.

4. Siapkan mekanisme untuk mengalokasikan pekerjaan ke anggota tim dan lacakpekerjaan yang dialokasikan sampai selesai.

5. Mengatur mekanisme komunikasi untuk alokasi dan penyelesaian pekerjaan, pelaporan dan penyelesaian masalah, eskalasi masalah, dll.

6. Pekerjaan pengembangan akan dimulai setelah lingkungan pengembangan dibuat, tetapi yang menjadi perhatian di sini adalah lingkungan pengujian. Lingkungan pengujian harus menjadi replika dari lingkungan pengembangan, meskipun pada tingkat yang diperkecil. Diperlukan server yang memiliki semua perangkat lunak yang diperlukan untuk database, server Web, dan tingkat menengah, serta beberapa mesin klien untuk melakukan pengujian. Mesin klien ini harus memiliki kit alat pengembangan perangkat lunak yang sama dengan yang dimiliki mesin klien pengembangan. Server harus memiliki data pengujian dengan fasilitas backup online sehingga dapat direset ke gambar pretest sebelum melakukan pengujian yang akan mengubah data pengujian.

Setiap kali komponen perangkat lunak perlu diuji, salinannya dipindahkan ke lingkungan pengujian ini. Komponen tersebut kemudian diuji. Setelah lulus dalam semua kasus uji, itu dihapus dari lingkungan ini, dan data uji diatur ulang ke gambar pra-uji. Untuk menguji aplikasi client-server, server harus memiliki database, dan perangkat lunak harus dimuat ke mesin klien. Penguji kemudian melakukan pengujian dari perangkat lunak klien dan data dari server. Untuk menguji aplikasi Web, perangkat lunak dan database harus berada di mesin server, dan penguji harus mengakses perangkat lunak menggunakan browser Web yang ditunjuk untuk menguji aplikasi. Untuk aplikasi mainframe, mesin front-end harus menggunakan perangkat lunak emulasi terminal, mengakses perangkat lunak dan data dari server, dan menguji perangkat lunak.

Lingkungan pengujian yang sebenarnya dijelaskan dalam rencana pengujian untuk proyek tersebut. Tidak jarang menggunakan lingkungan pengembangan itu sendiri sebagai lingkungan pengujian. Namun, ini bukan praktik terbaik. Lingkungan pengembangan akan memiliki perangkat lunak lain untuk membantu dalam debugging serta perangkat lunak lain yang tidak ada pada mesin pelanggan. Lingkungan pengujian harus semirip mungkin dengan lingkungan

target di lokasi pelanggan. Oleh karena itu, praktik terbaik adalah memiliki seperangkat mesin terpisah untuk lingkungan pengujian.

S. SKENARIO PENGUJIAN

Pengujian perangkat lunak independen dilakukan dalam dua skenario:

1. Pengujian proyek atau pengujian tertanam —Dilakukan sebagai bagian dari proyek pengembangan perangkat lunak untuk memastikan bahwa pekerjaan pengembangan bebas cacat. Hal ini bersamaan dengan pengembangan perangkat lunak.

2. Pengujian produk —Dilakukan pada produk perangkat lunak komersial yang tersedia untuk memastikan bahwa produk berfungsi tanpa cacat dalam berbagai skenario pelanggan. Atas permintaan khusus pelanggan, ini juga dapat diterapkan dalam pengembangan kontrak. Pengujian ini dilakukan setelah pengembangan perangkat lunak selesai.

T. PENGUJIAN PROYEK ATAU PENGUJIAN TERMASUK

Ketika perangkat lunak dikembangkan sebagai produk baik untuk dikirimkan ke klien tunggal atau dimaksudkan untuk digunakan di satu lokasi, pengujian produk tersebut dimulai segera setelah dimulainya pengembangannya. Artinya, unit perangkat lunak dikembangkan, dan kemudian unit itu dikenai pengujian yang ditentukan. Karena semakin banyak unit yang menyelesaikan pekerjaan pengembangan, unit-unit tersebut menjalani pengujian. Saat pekerjaan pengembangan berlangsung, pekerjaan pengujian juga berkembang, meskipun dengan jeda waktu antara pengembangan perangkat lunak dan pengujian. Saat pekerjaan pengembangan selesai, pekerjaan pengujian selesai segera setelahnya. Artinya, aktivitas pengujian tertanam dalam aktivitas pengembangan itu sendiri. Pada bagian berikut, jenis pengujian yang biasanya dilakukan dalam pengujian tertanam, selain verifikasi perangkat lunak, dibahas.

U. PENGUJIAN UNIT

Pengujian unit selalu dilakukan oleh orang yang menulis kode dan tambahan oleh rekan independen, menggunakan teknik pengujian kotak putih. Langkah-langkah berikut dilakukan dalam pengujian unit:

1. Pemrogram membangun sebuah komponen sesuai dengan dokumen desain untuk komponen tersebut, yang dialokasikan kepadanya oleh pemimpin proyek atau manajer proyek perangkat lunak. Pemrogram meninjau dan menguji komponen untuk memastikan bahwa ia melakukan fungsi yang dirancang. Pemrogram memberi tahu pemimpin proyek atau manajer proyek perangkat lunak bahwa komponen siap untuk pengujian unit.

2. Pemimpin proyek atau manajer proyek perangkat lunak mengidentifikasi rekan untuk melaksanakan pengujian unit, mengalokasikan pekerjaan melakukan pengujian unit kepada

orang yang diidentifikasi, dan menyediakan orang tersebut dengan kasus uji, pedoman, dan daftar periksa, jika ada, untuk melakukan pengujian unit.

3. Salinan kode tersedia untuk penguji.

4. Penguji memuat kode untuk komponen di lingkungan pengujian.

5. Penguji mengeksekusi semua kasus uji dan mencatat hasil aktual untuk masing-masing kasus test case, yang menunjukkan hasil lulus atau gagal.

6. Penguji menyerahkan log pengujian kepada pemimpin proyek atau manajer proyek perangkat lunak, yang mengatur perbaikan dari setiap cacat yang ditemukan selama pengujian dan meminta penguji untuk melakukan pengujian regresi dan menutup cacat.

7. Penguji melakukan pengujian regresi dan menutup cacat yang diperbaiki dan kemudian menyerahkan log pengujian kepada pemimpin proyek atau manajer proyek perangkat lunak untuk dimasukkan dalam catatan proyek.

8. Jika pengujian regresi menunjukkan cacat baru atau tidak terselesaikannya cacat sebelumnya, langkah 6 dan 7 diulang sampai semua cacat ditutup dengan memuaskan.

Platform pengembangan tertentu, terutama dalam pengembangan aplikasi Web, tidak mengizinkan melangkah melalui setiap baris kode menggunakan IDE atau debugger.Pengujian kotak putih dalam kasus seperti itu harus dicapai dengan mengakses aplikasi dari browser dan merancang kasus uji sedemikian rupa sehingga semua jalur dalam kode diuji secara komprehensif.

Karakteristik lain yang berbeda dari aplikasi Web adalah bahwa tahap pengujian unit mereka juga harus digunakan untuk pengujian sistem. Setiap kali komponen dalam aplikasi Web dibangun, komponen tersebut harus diuji tidak hanya untuk fungsi dan struktur tetapi juga dari semua browser Web target dan klien target. mesin. Jika aplikasi Web diuji hanya pada satu browser pada tahap pengujian unit, jumlah fungsionalitas yang perlu diuji pada tahap pengujian sistem akan sangat besar; hampir tidak mungkin untuk mengeksekusi semua kasus uji untuk memastikan bahwa aplikasi bekerja tanpa cacat pada semua mesin klien target dan browser Web.

V. TES INTEGRASI

Pengujian integrasi dilakukan baik sebagai satu kali (ketika semua integrasi selesai) atau secara bertahap (setiap kali satu unit perangkat lunak terintegrasi, integrasinya diuji sampai semua unit terintegrasi dan diuji). Pengujian kotak hitam digunakan untuk pengujian integrasi satu kali, dan pengujian kotak putih digunakan untuk pengujian integrasi mental tambahan. Pengujian integrasi harus mengadopsi pendekatan yang sama yang digunakan untuk mencapai pengujian integrasi produk oleh tim pengembangan. Ini harus didokumentasikan dalam dokumen rencana proyek untuk proyek tersebut.

Ada dua pendekatan untuk integrasi produk: pendekatan top-down dan pendekatan bottom-up. Pendekatan top-down untuk integrasi produk berlangsung sebagai berikut:

1. Komponen tingkat atas dari mana fungsionalitas produk akanmulai dikembangkan terlebih dahulu.

2. Komponen tingkat atas untuk modul produk dikembangkan selanjutnya, dan ini diintegrasikan dengan komponen tingkat atas produk.

3. Jika ada submodul, komponen tingkat atas untuk setiap submodul dikembangkan dan diintegrasikan dengan komponen tingkat atas dari modul masing-masing.

4. Setiap komponen dikembangkan dan diintegrasikan dengan komponen tingkat atas dari submodul atau modulnya masing-masing.

5. Proses ini berlanjut sampai semua komponen dikembangkan dan diintegrasikan parut dengan produk.

Dalam pendekatan top-down untuk pengujian integrasi, dua aspek integrasi akan diuji:

1. Kode antarmuka untuk mengintegrasikan modul dengan produk, submodul dengan modul, atau komponen dengan submodul atau modul diuji menggunakan teknik pengujian kotak putih.

2. Pengujian fungsional dan pemeriksaan navigasi selalu dilakukan dari komponen tingkat atas produk hingga bagian bawah produk dengan menggunakan teknik pengujian kotak hitam.

Pendekatan bottom-up untuk integrasi produk berlangsung sebagai berikut:

1. Setiap komponen pada level terendah dibangun dan diuji terlebih dahulu.

2. Ketika semua komponen submodul sepenuhnya dibangun dan diuji, komponen tingkat atas dari submodul dibangun, dan semua komponennya terintegrasi dengannya. Kemudian submodul tersebut diuji kemanjuran integrasinya.

3. Ketika semua submodul (atau komponen) dari sebuah modul telah selesai dibangun, komponen tingkat atas dari modul dibangun, dan submodul (atau komponen) tersebut diintegrasikan dengan komponen tingkat atas dari modul. Kemudian dilakukan pengujian integrasi untuk modul tersebut.

4. Setelah semua modul selesai dibuat, komponen tingkat atas produk dibuat, dan semua modul terintegrasi dengan produk. Kemudian dilakukan pengujian integrasi untuk produk tersebut.

Dalam pendekatan bottom-up untuk pengujian integrasi, pengujian dilakukan sebagai berikut:

1. Pengujian integrasi dilakukan setiap kali semua komponen submodul atau modul benar-benar dibangun dan terintegrasi.

2. Kode antarmuka untuk mengintegrasikan komponen dengan submodul atau modul diuji menggunakan teknik pengujian kotak putih.

3. Pengujian fungsional dan pemeriksaan navigasi selalu dilakukan untuk setiap submodul atau modul dengan menggunakan teknik pengujian kotak hitam, ketika semua komponen telah sepenuhnya dibangun dan terintegrasi dengan modul.

4. Pengujian fungsional dan pemeriksaan navigasi selalu dilakukan untuk setiap modul menggunakan teknik pengujian kotak hitam, ketika semua submodul (komponen) sepenuhnya dibangun dan diintegrasikan dengan modul.

5. Pengujian fungsional dan pemeriksaan navigasi selalu dilakukan untuk produk dengan menggunakan teknik pengujian kotak hitam, ketika semua modulnya sepenuhnya dibangun dan terintegrasi dengannya.

Pengujian integrasi dilakukan sebagai berikut:

1. Ketika integrasi (baik dalam pendekatan top-down atau bottom-up) selesai, pemimpin proyek atau manajer proyek perangkat lunak mengatur untuk tinjauan sejawat dan kemudian mengatur resolusi dan penutupan cacat yang ditemukan.

2. Salinan kode untuk submodul, modul, atau produk terintegrasi dipindahkan ke lingkungan pengujian oleh pemimpin proyek atau manajer proyek perangkat lunak.

3. Pemimpin proyek atau manajer proyek perangkat lunak mengidentifikasi penguji untuk melakukan pengujian integrasi yang diperlukan dan memberi penguji dengan kasus uji dan pedoman serta daftar periksa apa pun.

4. Penguji melakukan pengujian sesuai dengan kasus uji dan pedoman dan daftar periksa.

5. Penguji mencatat hasil aktual dan menentukan hasil lulus atau gagal untuk setiap kasus uji, yang juga dicatat.

6. Penguji menyerahkan log uji kepada pemimpin proyek atau manajer proyek perangkat lunak, yang mengatur penyelesaian cacat yang ditemukan.

7. Pemimpin proyek atau manajer proyek perangkat lunak memastikan bahwa semua cacat diselesaikan dengan memuaskan dan mengatur agar penguji melakukan pengujian regresi dan menutup cacat.

8. Penguji melakukan pengujian regresi dan menutup cacat. Log pengujian diserahkan kepada pemimpin proyek atau manajer proyek perangkat lunak untuk dimasukkan dalam catatan proyek.

9. Jika pengujian regresi menemukan cacat baru atau jika cacat sebelumnya tidak diselesaikan dengan memuaskan, langkah 6 sampai 8 diulang sampai semua cacat diselesaikan dengan memuaskan.

W. PENGUJIAN SISTEM

Pengujian sistem dilakukan untuk memastikan bahwa perangkat lunak bekerja pada semua sistem target yang dimaksudkan. Dalam aplikasi mainframe, pengembangan biasanya terjadi di lingkungan simulasi pada komputer pribadi berbiaya rendah. Setelah pengembangan selesai, perangkat lunak dimuat ke komputer mainframe dan diuji di lingkungan yang sebenarnya.

Dalam aplikasi client-server, pengujian sistem dilakukan pada semua versi sistem operasi yang dimaksud, seperti Windows 98, Windows 2000, Windows XP, dan Windows Vista. Ini juga dilakukan pada semua versi da tabase yang berlaku, seperti SQL Server 7, SQL Server 2000, SQL Server 2005, dll.

Dalam aplikasi Web, pengujian sistem mencakup pengujian pada berbagai sistem klien, seperti mesin berbasis Windows, mesin berbasis UNIX atau Linux, dan Mac. Aplikasi juga diuji pada semua browser Web populer, seperti Internet Explorer, Firefox, Google Chrome, dan lain-lain. Beberapa browser ini juga perlu digunakan dalam pengujian sistem. Negara-negara gabungan untuk pengujian sistem aplikasi Web sangat besar. Dalam aplikasi Web, sebagian besar pengujian sistem dilakukan selama tahap pengujian unit. Pengujian sistem perangkat lunak aplikasi Web membutuhkan banyak waktu untuk dilakukan.

X. RINGKASAN

Validasi menunjukkan konfirmasi atau pembuktian suatu klaim. Dalam konteks pengembangan perangkat lunak, validasi mengacu pada aktivitas yang dilakukan pada produk perangkat lunak untuk mengkonfirmasi bahwa semua fungsi yang dirancang (atau diperlukan) memang dibangun dan bekerja sesuai dengan spesifikasi asli (penggunaan yang dimaksudkan), bersama dengan fungsi implisit lainnya. untuk memastikan keselamatan, keamanan, dan kegunaan. Pengujian sistem dilakukan untuk memastikan bahwa perangkat lunak bekerja pada semua sistem target yang dimaksudkan. Dalam aplikasi mainframe, pengembangan biasanya terjadi di lingkungan simulasi pada komputer pribadi berbiaya rendah. Setelah pengembangan selesai, perangkat lunak dimuat ke komputer mainframe dan diuji di lingkungan yang sebenarnya.

Y. SOAL LATIHAN

1) Sebutkan 9 pengujian integrasi!
2) Sebutkan 2 skenario pengujian!Jelaskan!
3) Sebutkan 4 lingkungan uji!
4) Apa yang dimaksud Pengujian Intuitif?
5) Apa yang dimaksud dengan validasi?

BAB VI
KUALITAS KEANDALAN

A. BENCANA AKIBAT PERANGKAT LUNAK

Sistem semakin banyak yang dikendalikan oleh perangkat lunak belakangan ini. Penyebaran perangkat lunak dalam kehidupan kita begitu luas sehingga mungkin tidak berlebihan untuk mengatakan bahwa tidak ada perangkat keras yang tidak memiliki perangkat lunak pada saat ini. Barang-barang yang umum di rumah kita seperti televisi, mesin cuci, jam tangan, jam, sistem keamanan, dan sebagainya, dikendalikan oleh perangkat lunak. Bandara, pesawat terbang, roket, sistem senjata, mesin-mesin di pabrik, pusat perbelanjaan, anggap saja semuanya dikendalikan oleh perangkat lunak. Kita telah memberi label, terlepas apakah itu benar atau salah, pengembangan perangkat lunak telah disebut-sebut sebagai "teknologi tinggi", dan oleh karena itu kita berasumsi bahwa hal tersebut adalah hal yang sempurna, karena apa pun yang berteknologi tinggi harus sempurna. Banyak dari kita yang tidak mengerti atau tidak mencoba memahami seluk-beluk pengembangan perangkat lunak dan kita akhirnya bertumpu padanya, tetapi tenyata berbeda dengan faktanya, yaitu bahwa perangkat lunak rentan terhadap kesalahan, cacat, dan kegagalan seperti halnya perangkat keras tempatnya berada.

Telah banyak bukti di sektor manufaktur bahwa manusia adalah penyebab di balik sebagian besar kegagalan, dan penggunaan mesin menjadi solusi dalam mencapai kualitas. Faktor manusia sangat penting dalam pengembangan perangkat lunak, dan tidak berlebihan untuk mengatakan bahwa biaya tertinggi dalam pengembangan perangkat lunak dikeluarkan untuk manusia, oleh karena itu akibat ketergantungannya pada manusia, perangkat lunak berteknologi tinggi jauh menjadi lebih rentan terhadap kesalahan daripada perangkat keras manapun.

Internet penuh dengan berita tentang insiden bencana yang apabila ditelusuri akan ditemui bahwa penyebabnya adalah akibat kelemahan dari penggunaan perangkat lunak. Berikut ini adalah berita-berita yang beredar di internet terkait insiden tersebut:

- Pesawat terbang Lufthansa dengan nomor penerbangan 2904 mendarat dengan kecepatan 170 knot (314 km/jam) yaitu 20 knot (37 km/jam) lebih cepat dari kecepatan standar, sehingga pilot harus mengarahkan pesawat keluar dari landasan saat mencapai ujung. Sayap kiri menabrak tanggul dan kebakaran terjadi, api menyebar ke bagian ruang untuk penumpang. Akibatnya, kopilot dan satu penumpang tewas. Salah satu penyebabnya adalah akibat kerusakan dari perangkat lunak.

Gambar 6.1. Pesawat Lufthansa 2904

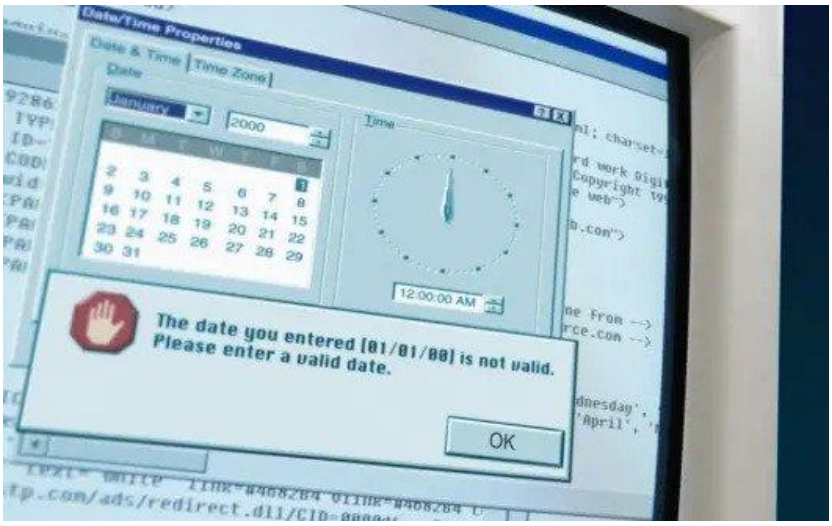
- Rudal Patriot MIM-104 dalam Operasi Badai Gurun Perang Teluk menyebabkan sebuah kecelakaan. Baterai di mana rudal ditembakkan beroperasi selama 100 jam, selama waktu itu jam internal sistem lepas sepertiga detik. Hal ini mengakibatkan rudal kehilangan target dengan jarak 600 meter. Akibat hal itu, rudal Scud tidak berhasil dicegat dan menghantam barak Angkatan Darat AS, dan menewaskan 28 tentara. Penyebabnya dilacak dan hasilnya mengarah ke kerusakan dari perangkat lunak.

Gambar 6.2. Rudal Patriot MIM-104

- Pada tahun 1999, misi untuk meluncurkan Polar Lander ke Mars gagal, karena perangkat lunak salah mengidentifikasi getaran di kaki kendaraan pada saat mendarat dan mematikan mesin pada ketinggian 40 meter dari permukaan yang mengakibatkan hilangnya Polar Lander tersebut.

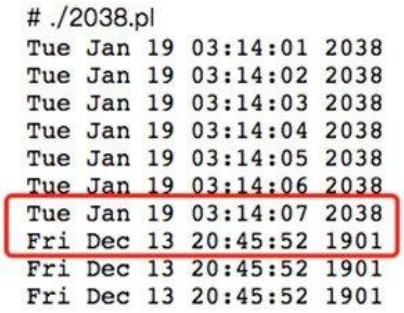
Gambar 6.3. Polar Lander

- Y2K kependekan dari Year 2000 adalah sebuah bug atau kesalahan komputer yang disebut juga sebagai Millenium Bug, menjadi pusat perhatian pada tahun 1990-an. Hal ini terjadi karena di masa-masa awal komputasi, untuk menghemat ruang disk dan memori selama eksekusi, tahun dimasukkan ke dalam sistem hanya menggunakan dua digit, dengan pemahaman bahwa setiap tahun empat digit dimulai dengan "19." Dengan datangnya tahun 2000, semua program ini harus diperbaiki, dengan perkiraan biaya 500 miliar dollar. Beberapa pihak menyelesaikan masalah Y2K ini dengan mengasumsikan setiap tahun yang dimasukkan sebagai "50" ke atas adalah abad ke-20 (1950 hingga 1999) dan setiap tahun yang dimasukkan sebagai "49" ke bawah adalah abad ke-21 (2000 hingga 2049). Masalahnya selanjutnya adalah, program-program ini jika tidak ditingkatkan teknologinya dan ditulis ulang, masalah serupa akan terjadi lagi pada tahun 2049 dengan indikasi bila program ini akan tetap berjalan pada tahun 2049.



Gambar 6.4. Millenium Bug

- Masalah tahun 2038, yaitu berkaitan dengan sistem komputer yang menggunakan sistem operasi UNIX. UNIX menyimpan waktu sistem sebagai jumlah detik dari 1 Januari 1970 sebagai integer 32-bit. Jumlahnya akan mencapai maksimum pada 19 Januari 2038, dan pada hari itu jika masalah ini tidak diperbaiki, jam akan diatur ulang dan tahun akan dianggap sebagai bagian dari abad ke-20.



Gambar 6.5. Masalah Tahun 2038 di UNIX

Insiden-insiden tersebut hanyalah contoh dari banyak masalah dan bencana yang disebabkan oleh perangkat lunak yang rusak atau cacat. Masalah perangkat lunak dalam bisnis tidak dilaporkan secara luas, karena tidak menyebabkan hilangnya nyawa atau kerugian ekonomi yang besar. Contoh beberapa insiden dalam bisnis yang disebabkan oleh masalah dengan perangkat lunak adalah sebagai berikut:

- Sebuah bursa saham mengkomputerisasi operasinya. Dalam waktu yang singkat setelah mengetahui bahwa ada beberapa orderan "jual" yang malah menghasilkan orderan "beli", bursa saham tersebut mematikan sistem komputer dan memperbaiki kesalahan sebelum mengaktifkan operasi berbasis komputernya lagi.
- Rumah sakit dengan 1.400 tempat tidur, mengalami masalah pada modul penagihan yang tidak mencakup semua layanan yang diberikan kepada pasien pada saat menerima pembayaran. Rumah sakit kehilangan pendapatan substansial dalam enam bulan sebelum anomali ini ditemukan dan diperbaiki. Butuh audit investigasi untuk menemukan kecacatan perangkat lunak terkait dengan hal ini.

Banyak orang juga memiliki pengalaman frustasi yang sama dengan PC mereka sendiri, yang tiba-tiba hang atau crash tanpa alasan yang jelas sering disebabkan oleh masalah perangkat lunak. Perangkat lunak, meskipun tidak rusak karena usia atau aus karena penggunaan, masih memiliki masalah terkait dengan keandalan.

B. KEANDALAN PERANGKAT LUNAK

Atribut yang harus dimiliki produk agar dapat diklaim berkualitas, selain fungsi bebas cacat, adalah keandalan. Artinya, produk melakukan fungsinya sesuai rancangan dengan baik dan konsisten selama masa pakainya. Jika sebuah mobil seharusnya memiliki masa manfaat hingga 100.000 mil, mobil dikatakan andal jika terus mengangkut lima orang sejauh 161.000 kilometer tanpa memerlukan perombakan atau perbaikan besar-besaran. Usia perangkat lunak tergantung pada usia perangkat keras yang menjalankannya. Hari-hari ini, perangkat keras menjadi usang dalam waktu tiga tahun (pengecualian bagi mainframe yang mahal), dan pemeliharaan mungkin

dilakukan selama lima tahun lagi. Selama masa pakai perangkat keras delapan sampai sepuluh tahun, banyak patch yang mungkin diperlukan untuk ditambahkan ke sistem perangkat lunak, seperti sistem operasi, sistem manajemen basis data, perangkat lunak antivirus, dan sebagainya. Patch ini dapat menyebabkan perangkat lunak gagal. Dengan demikian, keandalan produk perangkat lunak tergantung pada keandalan atau stabilitas konfigurasi sistem yang berfungsi.

Keandalan perangkat lunak didefinisikan sebagai kemungkinan bahwa perangkat lunak akan berfungsi tanpa kegagalan dan cacat, untuk jangka waktu tertentu dalam lingkungan tertentu. Keandalan berarti pelayanan tingkat kinerja yang bebas cacat yang konsisten selama masa pakai produk.

Produk perangkat lunak memang memiliki masa pakai yang terbatas, yang berarti durasi produk digunakan sebelum perlu diganti. Selama masa pakai produk, perbaikan perangkat lunak mungkin diperlukan karena alasan berikut:

(1) untuk memperbaiki cacat yang ditemukan selama penggunaan atau yang timbul sebagai akibat dari perubahan konfigurasi sistem seperti pembaruan pada sistem operasi dan utilitas lainnya,
(2) untuk meningkatkan fungsionalitas yang diperlukan karena lingkungan bisnis yang berubah,
(3) untuk memodifikasi fungsionalitas untuk memenuhi bisnis atau perubahan lingkungan.

Penggantian produk mungkin diperlukan ketika suatu produk membutuhkan perubahan besar dalam fungsi atau ketika ada perubahan paradigma dalam lingkungan, seperti adanya lonjakan kebutuhan akan internet. Kondisi seperti ini adalah kejadian yang jarang terjadi, dan oleh karena itu dapat diperkirakan bahwa suatu produk akan berada dalam pemeliharaan perangkat lunak untuk waktu yang lama.

Pemeliharaan perangkat lunak karena modifikasi atau peningkatan fungsional tidak serta merta membuat produk langsung dapat diandalkan. Namun, pemeliharaan perangkat lunak apa pun yang dilakukan untuk memperbaiki cacat yang ditemukan saat perangkat lunak dalam produksi tidak berarti juga bahwa produk tersebut sudah tidak dapat diandalkan. Tetapi jika semua kegiatan jaminan kualitas atau quality assurance (QA) telah dilakukan, bagaimana mungkin masih ada cacat di dalam produk perangkat lunak? Cacat mengintai suatu produk karena berbagai alasan, seperti berikut ini:

- Mungkin ada kekurangan dalam spesifikasi yang ditentukan untuk produk.
- Desain perangkat lunak itu sendiri mungkin cacat.
- Konstruksi produk perangkat lunak mungkin saja rusak.
- Semua aktivitas QA mungkin tidak dilakukan, atau bisa saja dilakukan tanpa ketekunan atau standar yang dibutuhkan.
- Fakta secara umum bahwa pengujian dan pengungkapan cacat hingga 100% terbukti tidak praktis, karena beberapa cacat masih tetap ada.
- Semua produk perangkat lunak dikembangkan menggunakan platform pengembangan, dan platform itu mungkin menyimpan beberapa cacat yang muncul hanya ketika serangkaian kondisi tertentu muncul.
- Perubahan konfigurasi sistem seringkali menimbulkan kondisi yang tidak terduga yang menyebabkan produk tidak berfungsi.

- Pembaruan dan patch pada perangkat lunak sistem seperti sistem operasi dan utilitas lain dapat menyebabkan produk tidak berfungsi.

Keandalan dicapai dengan membangun produk perangkat lunak sehingga tidak tergantung (atau tergantung pada tingkat minimum) pada shared libraries sistem operasi, yang kemungkinan akan diubah atau diperbarui selama periode waktu tertentu. Meminimalkan penggunaan alat pihak ketiga atau code libraries adalah praktik lain yang membantu menanamkan keandalan dalam suatu produk. Melakukan semua aktivitas QA pada suatu produk selama tahap pengembangannya adalah suatu keharusan untuk menghasilkan produk perangkat lunak yang andal. Keandalan produk perangkat lunak dikonfirmasi melalui pengujian perangkat lunak secara ekstensif. Ketika produk perangkat lunak dalam proses produksi, maka harus dipantau keandalannya menggunakan data berikut:

- Tingkat kegagalan: Ditentukan sebagai jumlah kegagalan selama periode waktu tertentu, seperti kegagalan per bulan kalender atau kegagalan per kuartal tahun.
- Downtime karena kegagalan perangkat lunak: Dinyatakan sebagai jumlah jam-jam produk tidak dapat digunakan. Lebih sering daripada tidak, kegagalan perangkat lunak tidak menyebabkan produk dimatikan. Terkadang kegagalan dapat menyebabkan fungsionalitas tertentu menjadi tidak tersedia. Downtime perlu dihitung meskipun hanya satu fungsi yang tidak tersedia.
- Pengeluaran yang dikeluarkan untuk perbaikan: Adalah normal untuk mendedikasikan beberapa personel untuk pekerjaan pemeliharaan, terlepas dari terjadinya kegagalan. Hal ini merupakan biaya tetap, konstan sepanjang umur produk. Sumber daya tambahan mungkin diperlukan untuk memperbaiki cacat dan untuk menangani kerusakan sistem ketika sumber daya khusus tidak tersedia atau tidak cukup untuk melakukan pemeliharaan yang diperlukan. Oleh karena itu, semua biaya yang dikeluarkan untuk pemeliharaan perangkat lunak perlu dipertimbangkan. Biaya per cacat (total pengeluaran pemeliharaan perangkat lunak untuk memperbaiki cacat dibagi dengan jumlah cacat yang diperbaiki) biasanya dihitung, dan nilai ini akan terus dipantau.

Nilai untuk semua data tersebut di atas dipantau menggunakan rata-rata bergerak dan analisis tren untuk menentukan analisis tingkat kegagalan.

C. PENYEBAB KEGAGALAN PERANGKAT LUNAK

Kegagalan perangkat lunak dapat disebabkan oleh cacat yang melekat pada perangkat lunak, seperti kesalahan, ambiguitas, kelalaian, asumsi yang salah, salah interpretasi persyaratan atau spesifikasi, tidak berfungsinya alat pihak ketiga, verifikasi dan validasi yang tidak memadai, dan penggunaan yang salah atau tidak terduga. Berbagai jenis kesalahan perangkat lunak dapat menyebabkan masalah keandalan, seperti yang dibahas di bagian berikut.

1) Kesalahan Desain Perangkat Lunak

- Miskinnya arsitektur
- Masalah penanganan pengecualian
- Masalah ketertelusuran spesifikasi atau persyaratan
- Salah tafsir spesifikasi atau persyaratan
- Penggunaan alat pihak ketiga tanpa mengujinya secara memadai

- Asumsi yang salah mengenai pengguna akhir atau platform target
- Verifikasi dan validasi desain yang tidak memadai
- Ukuran bidang tidak memadai, seperti masalah Y2K
- Perubahan lingkungan, seperti konversi euro

Cara terbaik untuk mencegah kesalahan desain perangkat lunak adalah dengan menerapkan kegiatan verifikasi dan validasi dengan rajin untuk desain perangkat lunak.

2) Coding atau Kesalahan Konstruksi

- Kelalaian
- Langkah yang terlewatkan
- Kesalahan kelalaian dan komisi
- Kode tidak efisien
- Kode ambigu
- Interpretasi perbandingan yang tidak tepat karena algoritma yang salah untuk perbandingan data
- Masalah presisi numerik

Pencegahan kesalahan konstruksi perangkat lunak dapat dicapai melalui pedoman pengkodean yang terdefinisi dengan baik, diikuti dengan verifikasi perangkat lunak dan pengujian yang ketat.

3) Masalah Jaminan Kualitas

- Verifikasi perangkat lunak tidak memadai
- Validasi perangkat lunak tidak memadai
- Tidak menggunakan penguji perangkat lunak khusus
- Tidak memverifikasi hasil tes
- Tidak mengukur kualitas produk perangkat lunak
- Tidak memiliki departemen QA

Memang benar bahwa aktivitas QA tidak dapat membangun kualitas menjadi produk secara langsung, tetapi tentu saja menyediakan lingkungan yang kondusif untuk menghasilkan perangkat lunak berkualitas tinggi yang bebas dari cacat. Kegiatan QA yang penting, selain verifikasi dan validasi, adalah definisi dan peningkatan proses perangkat lunak, pengukuran kualitas perangkat lunak, analisis dan peningkatan, dan definisi standar dan pedoman untuk kegiatan rekayasa perangkat lunak dan peningkatan rutinnya.

Banyak organisasi melihat departemen QA sebagai pusat biaya dan tidak menyediakan dana, sumber daya, dan alat yang diperlukan untuk berfungsi secara efektif. Selain itu, manajemen senior juga tidak mengalokasikan waktu yang cukup untuk kegiatan QA. Seperti ada aturan tak tertulis bagi manajemen senior untuk lebih memihak tim pengembangan dalam konflik apa pun antara departemen QA dan departemen pengembangan.

4) Kegagalan Data

- Kesalahan data induk

- Data master yang ambigu
- Masalah presisi data
- korupsi data
- Masalah integritas dan konsistensi data

Data yang buruk atau salah adalah salah satu penyebab utama kegagalan perangkat lunak, dan kemungkinan kegagalan perangkat lunak sangat tinggi, terutama ketika data dimasukkan secara offline, kecuali jika data divalidasi secara menyeluruh dan dibersihkan dari semua data yang buruk. Data master yang berisi data referensi untuk membuat keputusan sangat penting untuk pengoperasian perangkat lunak yang bebas dari kesalahan. Dilaporkan bahwa sebuah kapal perang ditenggelamkan karena perangkat lunak radarnya mengidentifikasi rudal yang masuk sebagai objek yang ramah! Jelas, perangkat lunak mencocokkan atribut objek yang masuk dengan objek yang ramah yang terkandung dalam file data masternya dan menentukan bahwa rudal itu adalah objek yang ramah.

Masalah lain yang sering terjadi dalam kualitas data adalah mengimpor data dari aplikasi lain. Ketika data diimpor langsung dari satu database ke database lainnya, kontrol terprogram tidak berfungsi untuk memvalidasi data input dan memastikan kualitasnya. Saat mengimpor data, penting untuk selalu memastikan bahwa datanya bersih dan akurat. Jika tidak, ini menjadi salah satu jalan untuk menyuntikkan data buruk ke dalam sistem.

D. PREDIKSI KEANDALAN PERANGKAT LUNAK

Ada banyak pengukuran keandalan dan metrik prediksi dan model dalam literatur perangkat lunak. Standar 982 dari Institut Listrik dan Elektronika Insinyur IEEE Standard Dictionary of Measures untuk Menghasilkan Perangkat Lunak yang Handal menyediakan serangkaian tindakan yang komprehensif. Namun, kebanyakan dari semuanya terlalu teoritis dan rumit untuk dipahami dan diterapkan, atau membutuhkan banyak waktu untuk menghitung. Beberapa tindakan yang diyakini mudah untuk diturunkan dan dipantau adalah sebagai berikut:

1) Product Metrics

- Perangkat lunak, yaitu semakin besar ukurannya, semakin tidak dapat diandalkan perangkat lunak dan semakin besar kemungkinan mengandung kesalahan.
- Kompleksitas program menggunakan kompleksitas siklomatik dari metrik McCabe dan Halstead.
- Pengujian coverage metric, yaitu menghitung jumlah kode perangkat lunak yang dicakup oleh pengujian. Idealnya harus 100%, tetapi dalam praktiknya, kurang dari itu.

2) Manajemen Proyek

Kecermatan dibutuhkan untuk melakukan aktivitas manajemen proyek. Karena dimungkinkan untuk mengambil jalan pintas selama pelaksanaan proyek, ada kemungkinan bahwa perangkat lunak akan mempertahankan kesalahan residual. Berikut ini adalah aspek penting dari manajemen proyek yang dapat mempengaruhi kualitas dan keandalan produk perangkat lunak:

- Manajemen mutu
- Manajemen konfigurasi
- Manajemen proyek
- Semangat tim

3) Metrik Proses Pengembangan Perangkat Lunak

Kualitas produk perangkat lunak bergantung pada proses pengembangan perangkat lunak yang diadopsi dalam organisasi. Ketatnya standar, pedoman, dan kegiatan QA memiliki pengaruh langsung pada kualitas produk. Berikut ini adalah aspek penting dari proses pengembangan perangkat lunak yang dapat mempengaruhi kualitas dan keandalan produk perangkat lunak:

- Kedalaman proses
- Ketatnya standar dan pedoman
- Ketekunan pelaksanaan proses
- Metrik kegagalan
- Jumlah kegagalan dalam periode waktu, seperti seperempat tahun
- Rata-rata waktu antara kegagalan
- Berarti waktu untuk memperbaiki

Peringkat kualitas produk komposit atau composite product quality rating (CPQR), adalah metode untuk menghitung metrik kualitas produk perangkat lunak. Hal ini merupakan prediksi yang sangat baik untuk keandalan perangkat lunak. Saat CPQR turun dari nilai 5, keandalan produk perangkat lunak juga menurun. Saat CPQR naik ke arah 5, demikian juga keandalan produk perangkat lunak ini meningkat. Sangat disarankan untuk menggunakan CPQR untuk memprediksi keandalan produk perangkat lunak.

E. PENINGKATAN KEANDALAN PERANGKAT LUNAK

Jelas, keandalan produk perangkat lunak tergantung pada keempat dimensi kualitas perangkat lunak. Sangat penting bahwa spesifikasi produk perangkat lunak, desain perangkat lunak, konstruksi, dan kesesuaian kualitas semuanya dilakukan dengan ketekunan yang sama dan tunduk pada QA yang sesuai, untuk memastikan bahwa kualitas dibangun ke dalam produk pada setiap tahap perkembangannya. Tidak ada satu dimensi yang dapat diberikan kepentingan yang lebih besar atau lebih kecil daripada yang lain.

Tidak ada peluru perak untuk mencapai keandalan perangkat lunak, tetapi ada beberapa teknik yang dapat membantu mencapai tingkat keandalan perangkat lunak yang diinginkan. Faktanya, semua aktivitas QA dirancang untuk meningkatkan keandalan perangkat lunak, dan tidak ada yang dapat diabaikan. Spesifikasi produk perangkat lunak yang berkualitas memastikan bahwa suatu produk dibuat dengan fungsionalitas yang komprehensif dan lengkap, termasuk fungsionalitas utama dan fungsionalitas tambahan. Kualitas desain perangkat lunak memastikan bahwa spesifikasi diimplementasikan dengan cara yang kuat, dengan toleransi kesalahan bawaandan perlindungan dari penyalahgunaan. Kualitas konstruksi perangkat lunak memastikan bahwa desain perangkat lunak diimplementasikan dan produk dikembangkan sepenuhnya sesuai

dengan desainnya. Kesesuaian kualitas memastikan bahwa tiga dimensi kualitas lainnya memang sesuai dengan standar dan praktik terbaik yang tersedia untuk produk.

F. RINGKASAN

Kegagalan perangkat lunak dapat disebabkan oleh cacat yang melekat pada perangkat lunak, seperti kesalahan, ambiguitas, kelalaian, asumsi yang salah, salah interpretasi persyaratan atau spesifikasi, tidak berfungsinya alat pihak ketiga, verifikasi dan validasi yang tidak memadai, dan penggunaan yang salah atau tidak terduga. keandalan produk perangkat lunak tergantung pada keempat dimensi kualitas perangkat lunak. Sangat penting bahwa spesifikasi produk perangkat lunak, desain perangkat lunak, konstruksi, dan kesesuaian kualitas semuanya dilakukan dengan ketekunan yang sama dan tunduk pada QA yang sesuai, untuk memastikan bahwa kualitas dibangun ke dalam produk pada setiap tahap perkembangannya. Tidak ada satu dimensi yang dapat diberikan kepentingan yang lebih besar atau lebih kecil daripada yang lain

G. SOAL LATIHAN

1) Sebutkan beberapa contoh beberapa insiden dalam bisnis yang disebabkan oleh masalah dengan perangkat lunak !
2) Sebutkan aspek penting dari proses pengembangan perangkat lunak yang dapat mempengaruhi kualitas dan keandalan produk perangkat lunak!
3) Sebutkan aspek penting dari manajemen proyek yang dapat mempengaruhi kualitas dan keandalan produk perangkat lunak!
4) Sebutkan kesalahan perangkat lunak dapat menyebabkan masalah keandalan!
5) Apa saja yang perlu dipantau keandalannya dalam proses produksi perangkat lunak?Sebutkan!

BAB VII
KUALITAS PROSES

A. EVOLUSI KUALITAS PROSES

Kematangan suatu organisasi dalam hal kualitas tercermin dalam produknya, karena pelanggan fokus pada produk atau layanan akhir yang mereka bayar dengan uang, wajar jika penyedia fokus pada kualitas produk atau layanan mereka. Mereka mencoba menyelesaikannya "entah bagaimana caranya", memastikan bahwa produk akhir memiliki kualitas yang dapat diterima, dan mengirimkannya. Meskipun sebagian besar pelanggan tidak menyadari proses di mana produk atau layanan disiapkan dan disampaikan, proses ini sebenarnya berdampak pada pelanggan. Misalnya, jika standar kebersihan yang tepat tidak dipatuhi di restoran selama persiapan makanan, makanan bisa terkontaminasi dan kesehatan orang yang makan di restoran itu bisa terpengaruh. Ketika produk makanan disiapkan dan dikemas di pabrik, praktik kebersihan makanan memainkan peran penting dalam menjaga konsumen aman dari penyakit.

Contoh ini dengan jelas menggambarkan pentingnya kualitas proses. Untuk produk tertentu, bagaimanapun, tidak mungkin untuk mengujinya untuk memastikan kualitas terjamin. Misalnya, bola lampu: tidak mungkin untuk memeriksa vakum atau menentukan jumlah gas inert di dalam selubung kaca tanpa memecahkan kaca dan bahkan bola lampu itu sendiri. Poros di mobil yang ditempa dan dikuatkan, tidak dapat diuji secara meyakinkan untuk kekuatan yang tepat.

Pernyataan bahwa proses pembuatan suatu produk sama pentingnya dengan produk itu sendiri memunculkan pepatah “kualitas tidak dapat ditambahkan, tetapi harus dibangun.” Pernyataan inilah yang menyebabkan konsep kualitas berkembang dari pengendalian kualitas ke aktivitas jaminan kualitas yang lebih komprehensif saat ini, dengan organisasi mengembangkan proses dan prosedur yang memfasilitasi spesifikasi dan desain produk dengan kualitas yang lebih tinggi. Seiring berjalannya waktu, konsep manajemen mutu total menekankan pada aspek proses penjaminan mutu, mengalihkan fokus dari pengendalian mutu produk menjadi pembuatan produk bermutu.

Pengembangan perangkat lunak memperkenalkan dimensi lain. Produk di industri lain melakukan atau tidak melakukan apa yang diharapkan, mereka tidak pernah melakukan antisipasi tentang hal yang tidak diharapkan untuk dilakukan. Entah berfungsi atau tidak. Misalnya, sebuah mobil berjalan atau tidak pasti tidak terbang! Namun, produk perangkat lunak selain berfungsi atau tidak, dapat melakukan operasi jahat yang tidak diharapkan atau tidak dimaksudkan untuk dilakukan. Contohnya adalah alat akselerasi unduhan. Kita mengharapkan alat untuk mempercepat tingkat pengunduhan, tetapi terkadang alat ini mengumpulkan informasi penggunaan Internet kita dan mengirimkannya ke pihak pengembang alat.

Untuk alasan ini, jaminan kualitas mendapat begitu banyak perhatian karena penekanannya pada kualitas proses sehingga Organisasi Internasional untuk Standardisasi atau International Organization for Standardization (ISO) merilis seri standar ISO 9000 pada tahun

1994, dengan fokus pada aspek proses ini. Industri pengembangan perangkat lunak menerima standar dengan antusias. Pelepasan standar ISO diikuti oleh Capability Maturity Model (CMM®) dari Institut Rekayasa Perangkat Lunak University pada tahun 1998. CMM® berfokus murni pada aspek pengembangan perangkat lunak, sedangkan ISO berfokus pada kualitas proses secara umum. Industri pengembangan perangkat lunak siap mengadopsi CMM® dan kemudian Capability Maturity Model Integration (CMMI®). Baik ISO 9000 dan CMM® berpusat pada kualitas proses, tetapi mereka berbeda dalam pendekatan mereka untuk menilai kepatuhan proses organisasi. Sementara ISO menyertifikasi organisasi yang sesuai dengan seri standar ISO 9000, CMM® dan CMMI® menilai organisasi dan menilai kemampuannya pada salah satu dari lima tingkat kematangan. Namun, keduanya mendorong pendekatan yang digerakkan oleh proses untuk fungsi organisasi.

B. PROSES

Sebelum membahas kualitas proses, proses itu sendiri harus dipahami. Ketika kita mencoba membuat sesuatu, kita mengikuti serangkaian langkah untuk menghasilkan produk akhir. Setiap aktivitas dalam kehidupan mengikuti suatu proses, bahkan jika proses itu tidak didefinisikan atau didokumentasikan secara eksplisit. Untuk beberapa kegiatan, prosesnya harus benar-benar dipatuhi. Misalnya, saat menggunakan resep untuk memasak hidangan, prosesnya tepat, dan kebebasan apa pun yang diambil dengan proses tersebut kemungkinan akan menghasilkan hidangan dengan rasa yang buruk atau makanan yang terlalu matang atau kurang matang. Untuk kegiatan lain, jika beberapa langkah dilewati atau dilakukan dengan buruk, hasilnya masih dapat diterima, setidaknya untuk sementara. Misalnya, jika permukaan tidak disiapkan dengan benar sebelum dicat, cat mungkin terlihat baik-baik saja setelah baru diterapkan, tetapi kemungkinan akan terkelupas nanti.

Ketika datang ke proses organisasi, pentingnya definisi proses dan kepatuhan proses pertama kali diakui dalam industri proses aliran seperti manufaktur kimia (pembuat pupuk, obat-obatan, produk minyak bumi, plastik, dan sebagainya). Aliran proses manufaktur menggunakan mesin untuk menghasilkan produk, dan pekerja akan memastikan bahwa proses tersebut dipatuhi dan melakukan koreksi yang diperlukan ketika ada penyimpangan terhadap kepatuhan proses yang diketahui. Setiap penyimpangan dari proses, baik di atas atau di bawah batas yang diizinkan, mengakibatkan konsekuensi bencana dari produk yang tidak dapat digunakan yang dihasilkan.

Tidak ada kepentingan signifikan yang diberikan untuk kepatuhan proses dalam domain manufaktur diskrit, kecuali di bidang-bidang seperti treatment panas, pengisian vakum (dalam bola lampu, misalnya), komponen semikonduktor, dan sebagainya. Definisi proses dan kepatuhannya menjadi penting hanya ketika koreksi a produk tidak mungkin, seperti yang terjadi di industri proses. Di mana perbaikan produk yang cacat dimungkinkan, seperti halnya dalam manufaktur diskrit, kepatuhan proses tidak mendapatkan banyak signifikansi.

Meskipun demikian, ditemukan bahwa definisi proses dan dokumentasi memiliki beberapa manfaat bagi suatu organisasi, seperti berikut ini:

- Memberi para eksekutif senior wawasan tentang bagaimana pekerjaan dilakukan di lantai pabrik.
- Membantu analisis oleh para ahli di lapangan, dan dengan melakukan hal tersebut, analisis ini membantu menutup celah dalam proses serta meningkatkan kualitas dan produktivitas.
- Memfasilitasi perencanaan dan penjadwalan kerja yang lebih ketat.
- Membantu mengungkap peluang untuk otomatisasi dan meningkatkan proses manufaktur itu sendiri.
- Memfasilitasi pengembangan standar untuk metode kerja serta untuk komponen.

Menyadari manfaat yang ditawarkan definisi proses dan dokumentasi, industri manufaktur diskrit mulai menggunakan proses yang ditentukan. Hal ini sejalan dengan berkembangnya cabang teknik industri yang menggunakan prinsip dan teknik studi kerja, sehingga dapat diterima bahwa definisi dan dokumentasi proses meningkatkan kualitas dan produktivitas. Orientasi proses ini pertama kali disebut sebagai praktik manufaktur yang baik, sebuah istilah yang masih digunakan sampai sekarang.

Pengembangan perangkat lunak dianggap sebagai pekerjaan kreatif dan oleh karena itu dipandang tidak memerlukan kerangka kerja standar dan proses yang kaku. Awalnya, tidak ada aktivitas kontrol kualitas yang dilakukan pada perangkat lunak yang dikembangkan. Standar rekayasa perangkat lunak Institute of Electrical and Electronics Engineers dirilis hanya pada akhir 1980-an, meskipun pengembangan perangkat lunak telah ada selama bertahun-tahun, yang menunjukkan betapa lazimnya kepercayaan bahwa perangkat lunak adalah karya kreatif.

Seri standar proses ISO 9000 memberi industri pengembangan perangkat lunak dorongan yang sangat dibutuhkan untuk merangkul kerja berorientasi proses, yang sudah menjadi norma dalam industri manufaktur.

C. KUALITAS PROSES

Setelah definisi proses dan dokumentasi mulai memberikan wawasan tentang bagaimana pekerjaan dilakukan di lantai pabrik, mereka digunakan sebagai alat untuk meningkatkan kualitas dan produktivitas. Disadari bahwa sebagian besar masalah kualitas dapat diselesaikan dengan meningkatkan proses itu sendiri. Analisis cacat digunakan untuk mengisolasi cacat yang sering terjadi dan untuk melacak asal-usulnya dalam proses. Area proses di mana cacat yang sama berulang kali terjadi diubah (dibuktikan) untuk menghilangkan cacat tersebut. Ini telah dan masih dicapai dengan meningkatkan metode, menggunakan alat yang lebih baik, atau melatih personel. Kesesuaian dengan proses dipastikan melalui inspeksi dalam proses dalam produksi proses aliran, produksi batch, dan manufaktur massal dan melalui inspeksi tahap dalam manufaktur diskrit.

Hari ini, alih-alih membiarkan cacat terjadi dan menggunakan inspeksi dan pengujian untuk mengungkapnya setelah fakta, proses dikembangkan untuk menghasilkan produk bebas cacat sehingga cacat tidak disuntikkan ke dalam produk sejak awal. Berikut ini adalah tiga langkah untuk mencapai kualitas proses:

1. Definisi proses
2. Peningkatan proses
3. Stabilisasi proses

D. DEFINISI PROSES

Langkah pertama dalam definisi proses adalah menetapkan entitas dalam tanggung jawab organisasi untuk memperjuangkan definisi dan peningkatan proses. Beberapa organisasi menugaskan tanggung jawab ini kepada departemen penjaminan mutu, yang lain menugaskannya ke grup proses spesialis. Siapa pun yang dipercayakan dengan tanggung jawab ini untuk memastikan kerja yang digerakkan oleh proses dalam organisasi dimulai dengan melakukan definisi proses. Kemudian, departemen yang bertanggung jawab mengumpulkan saran untuk perbaikan, mengevaluasi manfaat dan biaya masing-masing, dan menerapkan ke dalam proses saran yang dianggap memenuhi syarat. Definisi sebenarnya dari setiap proses dilakukan oleh praktisi dalam organisasi (spesialis yang memiliki pengalaman bertahun-tahun di bidang yang prosesnya sedang didefinisikan), dengan fasilitasi (alat, panduan dokumentasi, koordinasi kegiatan penjaminan mutu, dan sebagainya) disediakan oleh kelompok proses.

Ada dua pendekatan untuk mendefinisikan proses:

1. Pendekatan top-down: cocok ketika organisasi masih baru dan proses sedang diatur.
2. Pendekatan bottom-up: cocok ketika organisasi telah ada dan operasi telah dilakukan untuk beberapa waktu.

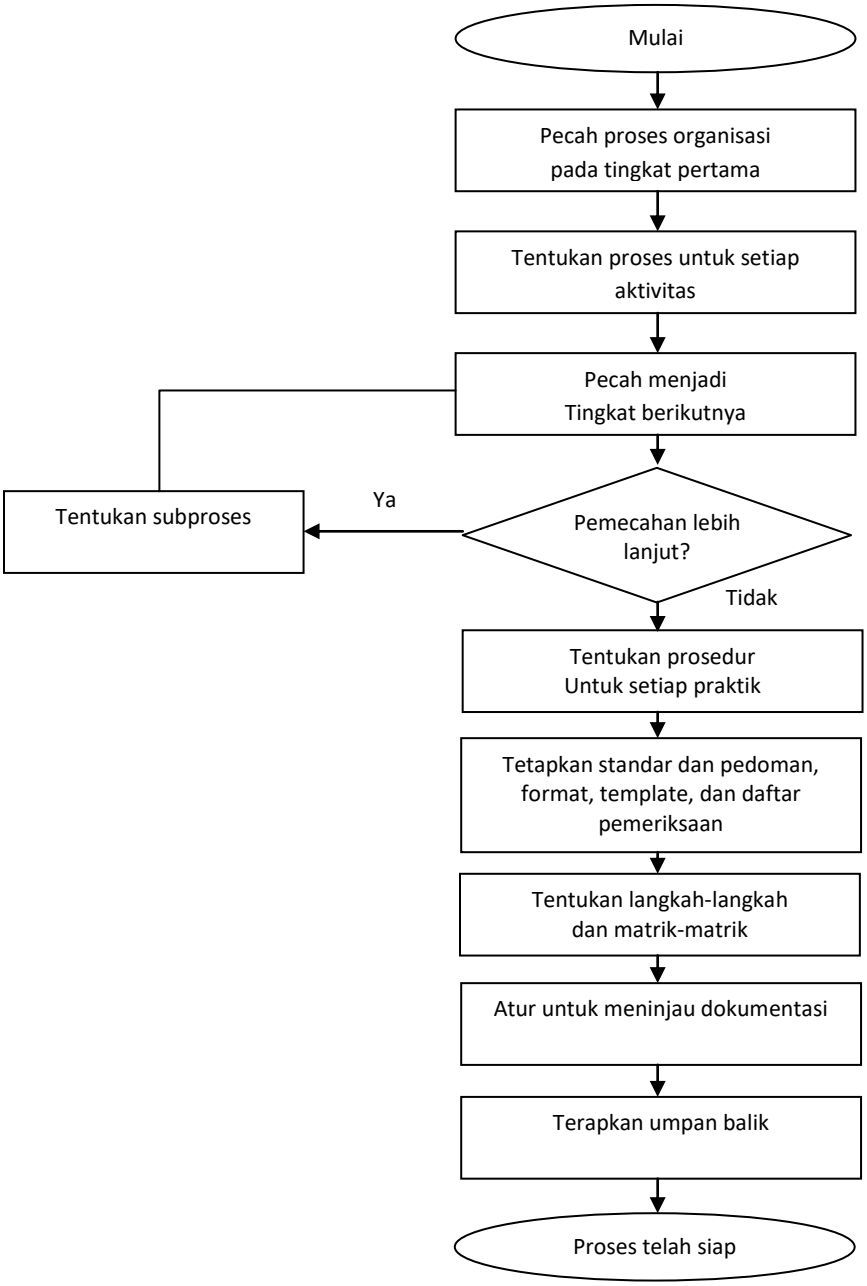
1) Pendekatan Top-Down untuk Definisi Proses

Pendekatan top-down untuk definisi proses terdiri dari langkah-langkah berikut:

1. Uraikan operasi organisasi berdasarkan fungsi. Tingkat pertama terdiri dari kegiatan utama operasi organisasi. Artinya, pengembangan perangkat lunak dipecah menjadi analisis kebutuhan, desain perangkat lunak, konstruksi, pengujian, dll pada tingkat pertama. Kemudian setiap aktivitas utama di tingkat pertama dipecah lagi ke tingkat berikutnya. Misalnya, desain perangkat lunak dipecah menjadi desain arsitektur, pemodelan data, desain basis data, desain antarmuka pengguna, desain laporan, dll. Lanjutkan memecah aktivitas sampai diputuskan bahwa perincian lebih lanjut tidak menambah nilai tambah apa pun
2. Tentukan proses untuk setiap aktivitas di tingkat perincian pertama.
3. Tentukan subproses untuk level berikutnya jika terdiri dari sublevel.
4. Tentukan prosedur untuk aktivitas yang tidak dipecah menjadi tingkat lebih lanjut.
5. Jika memungkinkan, tentukan standar dan pedoman untuk membantu praktiksi dalam mengikuti prosedur.
6. Tentukan format dan template untuk merekam dan menyajikan informasi dan untuk mencapai keseragaman di antara praktisi yang berbeda dalam menangkap, merekam, dan menyajikan informasi.
7. Tentukan daftar periksa untuk membantu praktisi dalam mematuhi prosedur dan melakukan kegiatan secara komprehensif dan menyeluruh.
8. Tentukan pengukuran dan analisis hasil dari proses yang ditentukan untuk menilai keefektifannya.

9. Atur peninjauan proses yang ditetapkan oleh praktisi dalam organisasi untuk memastikan bahwa itu mencerminkan kenyataan dan didefinisikan secara akurat.

Langkah-langkah ini digambarkan pada gambar berikut:



Gambar 7.1. Pendekatan Top-Down untuk Definisi Proses

2) Pendekatan Bottom-Up untuk Definisi Proses

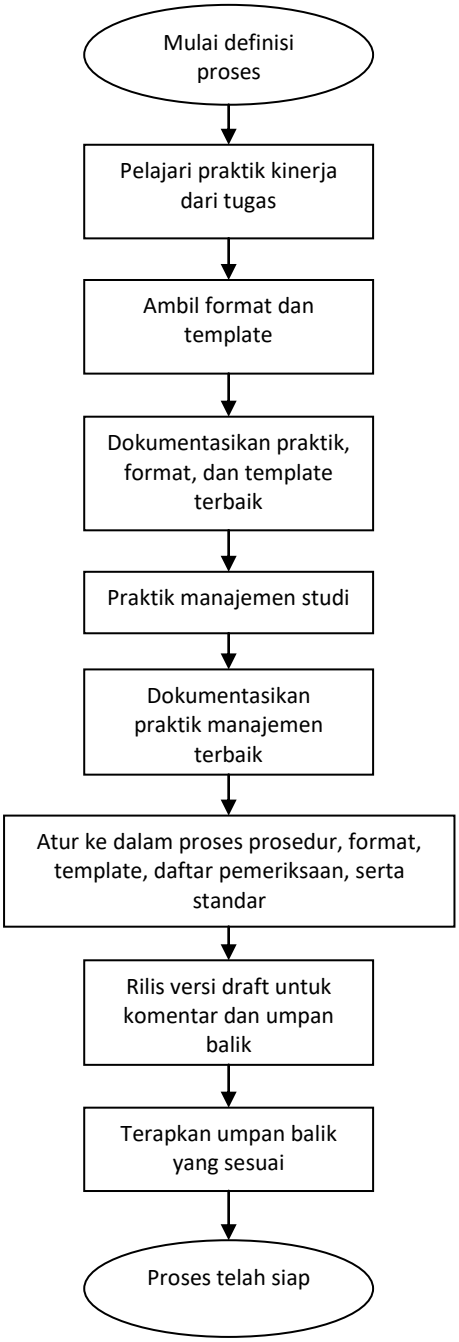
Pendekatan bottom-up untuk definisi proses terdiri dari langkah-langkah berikut:

1. Pelajari bagaimana praktisi melakukan tugas mereka, dan dokumentasikan informasi ini. Jelas, akan ada perbedaan di antara praktisi dalam cara kegiatan dilakukan. Oleh karena

itu, sertakan dalam dokumen proses praktik yang paling umum dan praktik orang-orang yang dikenal untuk menghasilkan kiriman dengan kualitas terbaik.

2. Menangkap format dan template yang digunakan oleh praktisi. Pelajari dan kembangkan format dan template baru yang menyertakan fitur terbaik dari setiap format dan template yang digunakan dalam organisasi.
3. Menangkap detail proses dari manajer proyek dan manajer senior, dan mendokumentasikannya, memusnahkan praktik terbaik.
4. Mengatur materi ke dalam proses, prosedur, format, daftar periksa, serta standar dan pedoman.
5. Meluncurkan versi draf proses organisasi, dan mengundang komentar dari semua orang yang terkait dalam organisasi.
6. Menganalisis umpan balik yang diterima yang berfungsi untuk meningkatkan kualitas atau produktivitas atau menyederhanakan pekerjaan. Setelah menerapkan umpan balik terpilih, lepaskan proses untuk implementasi di organisasi.

Langkah-langkah ini digambarkan pada gambar berikut:



Gambar 7.2. Pendekatan Bottom-Up untuk Definisi Proses

E. MEMBANGUN KUALITAS KE DALAM PROSES YANG DITETAPKAN

Setelah dokumen proses disiapkan, dokumen tersebut harus diperiksa oleh para ahli di bidangnya, baik internal maupun eksternal organisasi, sebelum dirilis untuk diimplementasikan. Para ahli ini akan mengevaluasi setiap langkah dan prosedur proses dan membandingkannya dengan praktik terbaik di industri. Evaluasi dan perbandingan ini menghasilkan dokumen analisis kesenjangan. Kelompok proses internal organisasi harus menganalisis setiap celah yang

ditemukan untuk kelayakannya untuk dijembatani dalam proses dan praktik organisasi. Analisis internal ini mungkin memerlukan investasi tambahan dalam peralatan, pelatihan untuk personel, perubahan metode kerja, dll. Kadang-kadang dimungkinkan untuk menerapkan praktik terbaik sepenuhnya; di lain waktu mungkin perlu dimodifikasi agar sesuai dengan lingkungan organisasi. Dalam beberapa kasus, organisasi mungkin perlu sepenuhnya menolak praktik terbaik jika tidak sesuai dengan lingkungannya. Sejalan dengan keputusan untuk menerapkan praktik terbaik yang disarankan, dokumen proses harus diperbarui dan diselesaikan. Dimasukkannya praktik terbaik dalam dokumentasi proses terlebih dahulu dan menyesuaikannya ke dalam praktik kedua memastikan bahwa kualitas dibangun ke dalam proses.

Kegiatan penjaminan mutu yang diperlukan kemudian dimasukkan ke dalam dokumen proses untuk memastikan bahwa proses yang ditentukan tunduk pada penjaminan mutu selama praktik. Kegiatan ini dapat mencakup tinjauan, pengujian, inspeksi, dan audit. Juga termasuk ukuran dan metrik yang diperlukan untuk menilai kinerja proses. Kegiatan ini membangun kualitas ke dalam proses.

F. MENYESUAIKAN PROSES DENGAN MODEL PROSES

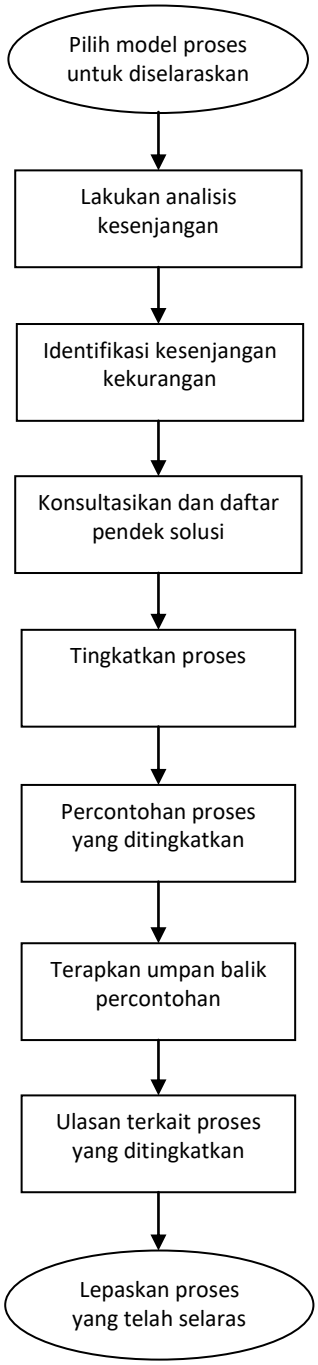
Menyelaraskan proses organisasi dengan model proses seperti ISO 9000 atau CMMI® menjadi penting, karena semakin banyak pelanggan yang menuntut sertifikat kepatuhan atau peringkat jatuh tempo sebagai prasyarat untuk berpartisipasi dalam proses penawaran untuk kontrak outsourcing. Namun, proses organisasi yang didefinisikan secara komprehensif dan mencakup praktik terbaik industri cukup untuk memenuhi persyaratan model apa pun, karena tujuan utama dari semua model adalah untuk memastikan bahwa organisasi menggunakan praktik terbaik industri dalam fungsinya dan dalam menghasilkan hasil yang berkualitas. Setelah dipastikan bahwa proses organisasi menggunakan praktik terbaik industri, semua yang perlu dilakukan adalah memastikan bahwa model yang dipilih dan proses organisasi sinkron satu sama lain dan bahwa proses organisasi memenuhi atau melampaui persyaratan tujuan model.

Langkah-langkah berikut diperlukan untuk menyelaraskan proses organisasi dengan model seperti itu:

1. Mempelajari persyaratan model, terutama tujuan yang harus dipenuhi oleh proses dan praktik organisasi.
2. Lakukan analisis kesenjangan antara kebutuhan model dan proses organisasi. Kesenjangan bisa positif, artinya praktik organisasi melebihi persyaratan model, atau kekurangan, artinya proses organisasi tidak memenuhi persyaratan model.
3. Hitung semua kesenjangan kekurangan (contoh dalam proses organisasi yang tidak memenuhi tujuan model) dalam dokumen analisis kesenjangan.
4. Melakukan serangkaian konsultasi dengan praktisi dan manajemen organisasi tentang kesenjangan dan cara untuk menjembataninya.
5. Pilih solusi alternatif yang paling cocok untuk menjembatani kesenjangan dan menerapkannya dalam proses.

6. Cobalah proses-proses yang telah diperbaiki pada basis percontohan selama pelaksanaan beberapa proyek.
7. Menerapkan umpan balik dari implementasi percontohan ke dalam proses.
8. Atur untuk ditinjau oleh praktisi di organisasi.
9. Melepaskan set proses baru untuk implementasi dalam organisasi.

Langkah-langkah ini, yang digambarkan pada gambar berikut, memastikan keselarasan proses dan praktik organisasi dengan model proses yang dipilih.



Gambar 7.3. Menyelaraskan Proses Organisasi dengan Model yang Dipilih

G. PENINGKATAN PROSES

Setelah organisasi menetapkan serangkaian proses, dan jika organisasi telah mencapai keselarasan dengan model proses seperti ISO 9000 atau CMMI®, kinerja proses harus dipantau terus-menerus dalam hal kinerja aktual vis-à-vis (face to face) yang diinginkan. pertunjukan. Pemantauan penting karena iklim organisasi dapat sering berubah. Beberapa alasan perubahan ini antara lain:

- Perubahan teknologi, yang menghasilkan paradigma baru untuk pengembangan perangkat lunak
- Penggunaan alat dan teknik yang lebih baik oleh pesaing untuk menghadirkan perangkat lunak berkualitas lebih tinggi dengan harga lebih murah
- Peraturan pemerintah untuk keamanan, kegunaan, keandalan, dll., yang mungkin mengharuskan untuk memberikan kualitas yang lebih baik
- Ketersediaan alat pengembangan baru, yang dapat menghemat uang atau meningkatkan kualitas
- Ketersediaan produk baru untuk back end, tingkat menengah, platform pengembangan, dan sebagainya

Hal tersebut hanyalah beberapa alasan mengapa organisasi didorong untuk meningkatkan proses mereka untuk memenuhi lingkungan yang berubah dan menjadi kompetitif di pasar. Oleh karena itu, peningkatan proses organisasi harus didekati secara disiplin, yaitu, dengan pendekatan yang digerakkan oleh proses.

Sebuah proses untuk meningkatkan proses organisasi harus didefinisikan, sama seperti proses organisasi lainnya. Sebuah proses untuk perbaikan proses harus mencakup bidang-bidang berikut:

1. **Pemicu untuk Peningkatan Proses:**

   Pemicu dapat didasarkan pada peristiwa atau durasi. Pemicu berbasis peristiwa dapat berupa laporan audit oleh auditor eksternal atau penilai eksternal atau laporan dari audit atau penilaian internal di seluruh organisasi. Ini juga mungkin alat baru yang berdampak pada proses, atau mungkin rilis standar baru atau versi baru dari model proses yang diadopsi organisasi yang berdampak pada operasi organisasi.

   Pemicu berbasis durasi terjadi sekali setiap kuartal, enam bulan, atau tahun atau pada awal tahun fiskal baru, misalnya. Pemicu ini digunakan untuk melakukan latihan untuk mengkonsolidasikan semua peluang peningkatan proses, menganalisisnya, dan memengaruhi peningkatan proses.
2. **Sumber Peluang untuk Perbaikan Proses:**

   Berbagai sumber informasi untuk perbaikan proses diidentifikasi. Sumber informasi ini dapat mencakup sumber internal, seperti saran dari anggota tim pengembangan, manajer, dan manajemen senior, atau sumber eksternal, seperti saran dari auditor eksternal atau penilai atau badan standar atau pemilik model proses.
3. **Prosedur untuk Menempatkan Permintaan Perbaikan Proses:**

Prosedur ini mencakup bagaimana menempatkan permintaan perbaikan proses, format atau template apa yang tepat, informasi apa yang perlu disertakan dalam permintaan, persetujuan apa yang diperlukan, dan sebagainya.

4. **Orang yang Berwenang yang Dapat Menempatkan Permintaan Perbaikan Proses:**

   Hal tersebut adalah daftar yang menunjukkan jenis permintaan yang dapat diajukan oleh berbagai orang dalam organisasi. Misalnya, pemrogram perangkat lunak dapat mengajukan permintaan perbaikan proses mengenai pedoman pengkodean atau praktik pengujian. Tidak tepat bagi seorang programmer untuk mengajukan permintaan untuk meningkatkan prosedur analisis cacat, misalnya. Artinya, seseorang paling cocok untuk mengajukan permintaan perbaikan di area tempat dia bekerja. Kriteria ini disebutkan dalam bagian ini.

5. **Prosedur untuk Menganalisis dan Menerima Permintaan Perbaikan Proses:**

   Bagian ini menjelaskan jenis analisis yang dapat dilakukan disalurkan pada permintaan perbaikan proses yang diterima oleh proseskelompok perbaikan. Badan yang berwenang untuk menyetujui atau menolakpermintaan peningkatan proses menganalisis permintaan dan menerimaatau menolaknya. Jika permintaan ditolak, pencetus permintaan adalahdiberitahu tentang penolakan dan alasan mengapa. Jika permintaan diterima, perbaikan dilaksanakan secara normal.

6. **Prosedur untuk Mengimplementasikan Perbaikan Proses:**

   Bagian inimenjelaskan cara memilih orang untuk menerapkan perbaikan yang disarankandalam dokumen proses, norma versi untuk dokumen proses, tinjauan yang akan dilakukan untuk dokumen proses yang ditingkatkan, dan otorisasi persetujuan.

7. **Prosedur Pelaksanaan Percontohan untuk Mendapatkan Umpan Balik Lapangan:**

   Sekali dokumen proses diperbaiki dan disetujui, mereka diterapkan pada beberapa proyek untuk mendapatkan umpan balik dari lapangan. Ini bagian menjelaskan prosedur seperti itu, termasuk cara memilih kandidat proyek, keringanan hibah yang diperlukan untuk menyimpang dari saat ini proses yang disetujui, mendapatkan umpan balik dari implementasi percontohan, mengumpulkan dan menganalisis umpan balik lapangan, dan sebagainya.

8. **Menerapkan Umpan Balik dari Lapangan dan Merilis Proses:**

   Setelah umpan balik dari lapangan diperoleh, itu harus diimplementasikan dalam proses. Bagian ini menjelaskan cara menerapkan umpan balik lapangan, mengatur tinjauan, mendapatkan persetujuan yang diperlukan, dan merilis perbaikan proses untuk implementasi. Ini adalah praktik normal untuk mempengaruhi hanya beberapa rilis per periode, seperti setiap kuartal, enam bulan, atau tahun.

H. STABILISASI PROSES

Stabilisasi proses diperlukan bagi organisasi untuk menghasilkan produk yang dapat diprediksi hasil. Namun, bukan berarti perbaikan proses tidak diperlukan. Hal ini hanya berarti bahwa suatu proses harus, pada umumnya, stabil. Perbaikan dipengaruhi berdasarkan pemicu, dan diterapkan untuk meningkatkan kualitas atau produktivitas atau untuk menyederhanakan pekerjaan. Stabilisasi proses menjadi mungkin, hanya setelah proses telah didefinisikan.

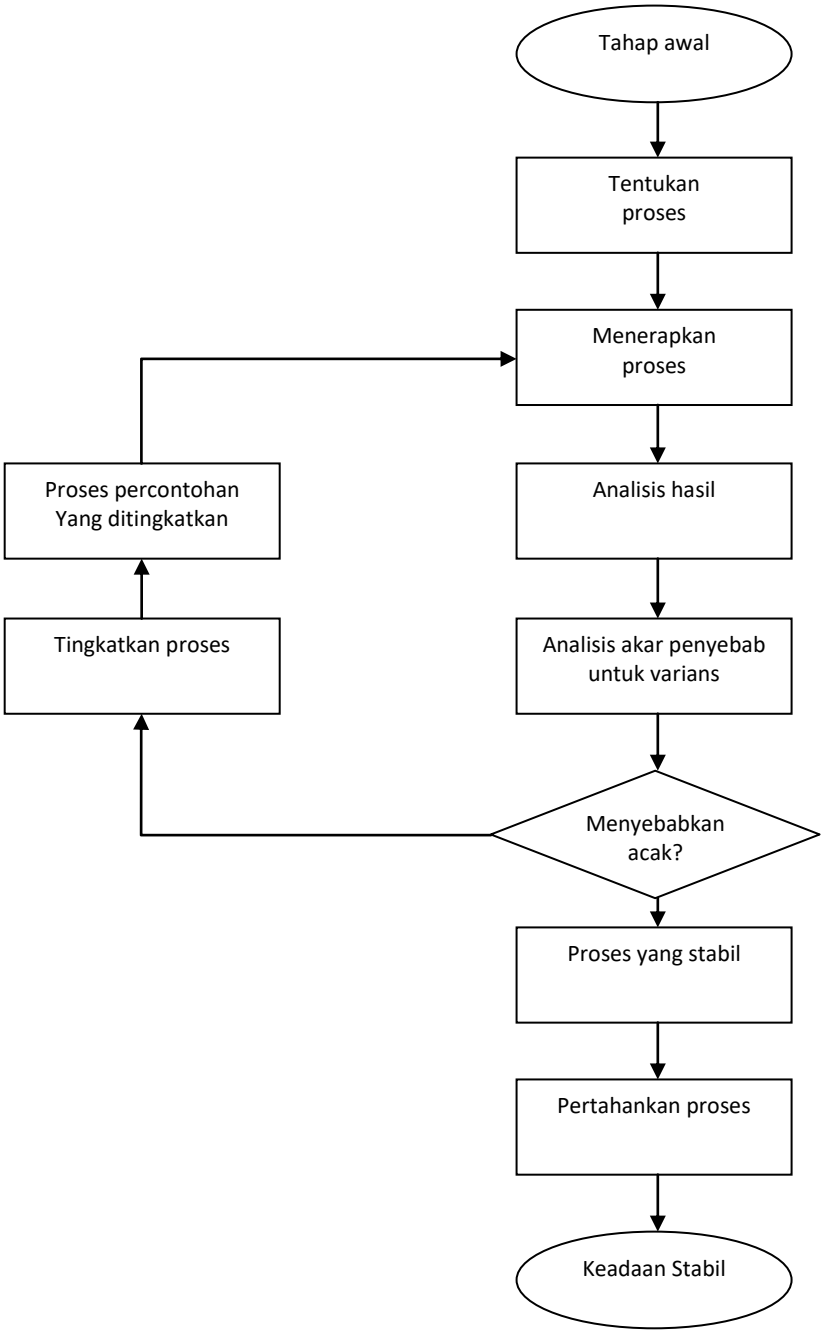
Sebuah organisasi biasanya melewati tahap-tahap berikut sebelum mencapai proses yang stabil:

1. **Tahap awal:** adalah saat organisasi baru dan baru dimulai operasinya, yang mencoba untuk membangun kelayakan komersial. Operasi dilakukan berdasarkan arahan pribadi pemilik, kepala eksekutif, dan manajemen senior.
2. **Mendefinisikan proses:** Setelah organisasi telah mencapai kelangsungan hidup komersial, mendefinisikan proses untuk melakukan operasi menggunakan pendekatan proses-driven.
3. **Menerapkan proses**: Proses yang ditentukan diimplementasikan dalam organisasi. Semua operasi dijalankan berdasarkan proses yang ditentukan, dan proses tersebut dilembagakan dalam organisasi.
4. **Mempertahankan proses:** Setelah proses diimplementasikan, proses tersebut dipantau dan ditingkatkan sesuai kebutuhan, berdasarkan pemicu peristiwa atau durasi, dengan menggunakan langkah-langkah berikut:
   a. **Analisis hasil operasi:** untuk setiap varians—Varians dapat bermanfaat jika, misalnya, menghasilkan kualitas atau produktivitas yang lebih tinggi. Varians yang tidak diinginkan menghasilkan lebih banyak cacat atau produktivitas yang berkurang, misalnya.
   b. **Melakukan analisis akar penyebab varians:** Ketika varians yang tidak diinginkan ditemukan, analisis akar penyebab dilakukan. Beberapa varians ini dapat murni karena kesalahan kebetulan. Bahkan dalam proses yang paling ketat dikontrol dan berbasis mesin, varian acak memang terjadi, dengan beberapa karena penyebab yang dapat ditentukan. Analisis akar penyebab memisahkan varians yang tidak diinginkan menjadi kesalahan kebetulan dan penyebab yang dapat ditentukan dan kemudian mengungkap penyebab sebenarnya di balik setiap varians.

      Varians yang tidak diinginkan karena penyebab yang dapat ditentukan dianalisis untuk menentukan apakah mereka disebabkan oleh cacat proses atau cacat lainnya. Jika karena cacat proses, maka perbaikan harus dilakukan dalam proses yang ditentukan. Untuk varians yang tidak diinginkan karena penyebab yang dapat diatribusikan pada proses yang rusak, perbaikan harus dilakukan untuk menutup celah dalam proses sehingga varians tersebut tidak terulang.
   c. **Proses percontohan yang ditingkatkan:** Proses yang ditingkatkan diimplementasikan dalam beberapa proyek untuk mengamati kemanjuran perbaikan. Jika hasilnya tidak menghasilkan perbaikan yang diinginkan, langkah b dan c diulang sampai perbaikan yang diinginkan tercapai berdasarkan implementasi percontohan.
   d. **Implementasikan proses yang ditingkatkan:** Setelah implementasi percontohan dari proses yang ditingkatkan menunjukkan tingkat perbaikan yang diinginkan, proses tersebut melewati prosedur normal untuk mengimplementasikan suatu proses dalam organisasi.

5. **Proses yang stabil:** Ketika sebagian besar atau semua varians disebabkan oleh penyebab acak (kebetulan), maka proses dianggap stabil.

Gambar berikut menggambarkan stabilisasi proses. Setelah proses stabil, statistik teknik kontrol kualitas seperti diagram kontrol dapat digunakan untuk memantaunya.



Gambar 7.4. Stabilisasi Proses

I. PROSES PENGEMBANGAN PERANGKAT LUNAK

Proses organisasi terdiri dari empat jenis proses dasar:

a. Proses rekayasa perangkat lunak:

Proses ini menentukan bagaimana deliverable dibangun. Mereka biasanya terdiri dari proses untuk manajemen kebutuhan, desain perangkat lunak, konstruksi, dan penyebaran.

b. Proses penjaminan kualitas:

Proses ini menentukan bagaimana kualitas itu dibangun ke dalam kiriman dan juga memastikan bahwa itu benar-benar dibangun. Mereka biasanya terdiri dari verifikasi, validasi, inspeksi, pengukuran dan analisis, dan audit.

c. Proses manajemen:

Proses-proses ini menentukan bagaimana semua proses lainnya proses dikelola. Mereka biasanya berisi proses manajemen proyek, termasuk akuisisi proyek, inisiasi proyek, estimasi perangkat lunak, perencanaan, manajemen konfigurasi, manajemen kualitas, manajemen kerja, manajemen sumber daya, pemangku kepentingan.manajemen harapan, dan penutupan proyek.

d. Proses pendukung:

Proses ini menentukan bagaimana proses lainnya didukung oleh organisasi. Mereka terdiri dari jaringan kerja dan proses administrasi sistem, proses sumber daya manusia, proses manajemen subkontraktor, proses manajemen fasilitas, dll.

J. KOMPONEN DARI PROSES

Proses adalah definisi keseluruhan dari aktivitas organisasi utama. Ini adalah sebuah jaringan prosedur yang saling terkait untuk melakukan suatu aktivitas dan tingkat atas dokumen di mana dokumen lain memberikan rincian kegiatan.

Sebuah proses adalah kumpulan dari komponen-komponen berikut:

a. Prosedur:

Prosedur adalah instruksi langkah demi langkah untuk melakukan subaktivitas dari suatu proses. Contoh prosedur termasuk prosedur perencanaan proyek, prosedur estimasi perangkat lunak, prosedur audit akhir fase, prosedur pelaporan kemajuan, dll.

b. Standar dan pedoman:

Hal ini menentukan cara umum untuk membangun artefak dalam organisasi sehingga keluarannya seragam di seluruh organisasi. Sedangkan prosedur membawa keseragaman dalam melakukan suatu kegiatan, standar dan pedoman membawa keseragaman dalam kiriman. Contoh standar dan pedoman adalah pedoman desain layar, pedoman desain basis data, standar penamaan, pedoman pengkodean, standar analisis cacat, dll.

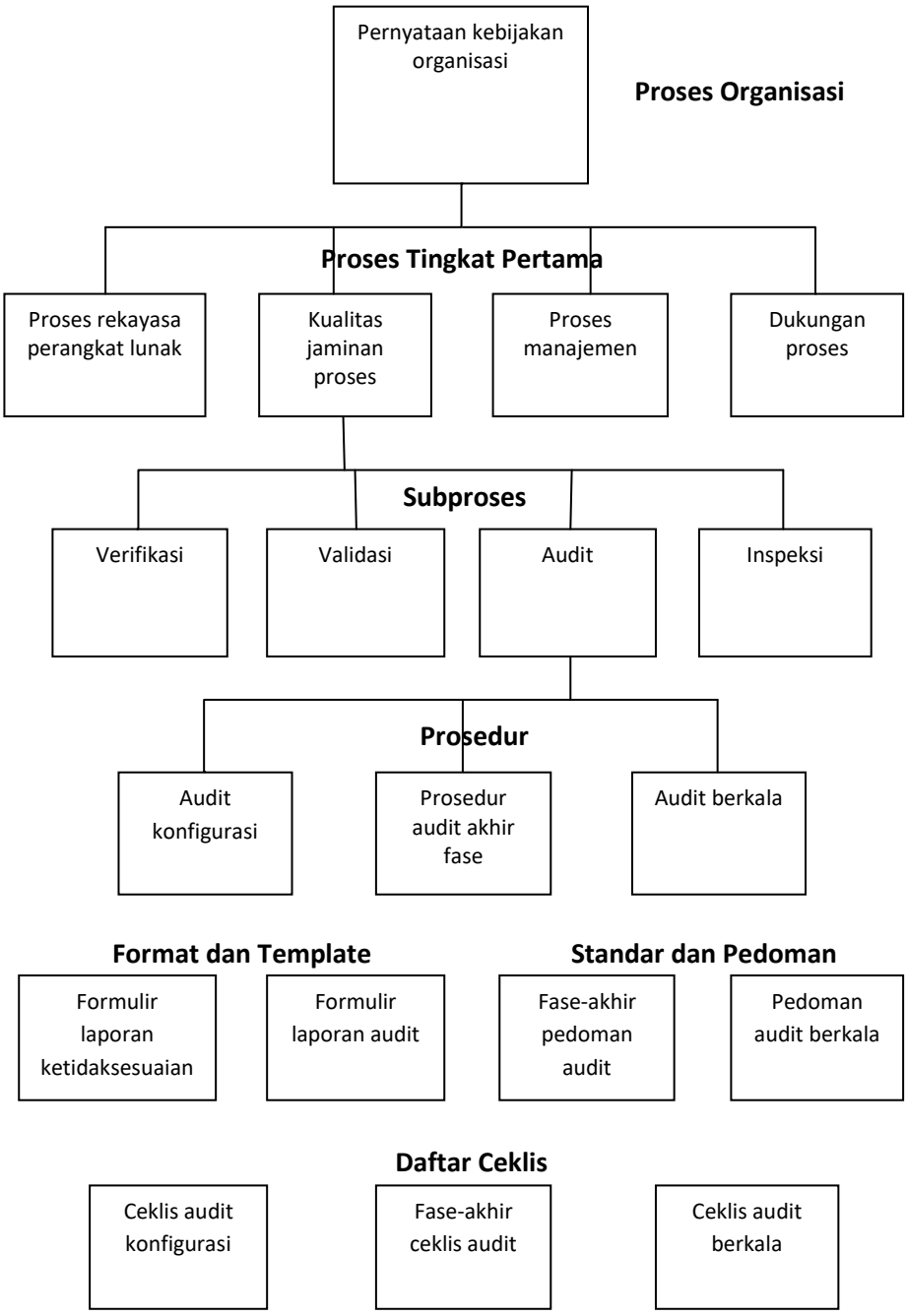
c. Format dan template:

Hal ini memfasilitasi cara yang seragam untuk menangkap, merekam, dan menyajikan informasi. Contoh format dan templat termasuk templat presentasi estimasi, formulir laporan ketidaksesuaian, formulir tinjauan cacat, catatan permintaan estimasi, templat rencana manajemen proyek, dll.

d. Daftar Periksa:

Hal ini membantu orang yang melakukan suatu kegiatan untuk melaksanakan tugas sepenuhnya dan membantu peninjau untuk memastikan kelengkapan tinjauan. Daftar periksa berisi sejumlah item, dengan spasi di samping masing-masing untuk menunjukkan "ya", "tidak", atau "tidak berlaku" karena setiap poin diselesaikan atau ditinjau. Ketika suatu kegiatan selesai, daftar ini harus dirujuk untuk memastikan bahwa semua poin telah ditangani.

Hirarki dari kumpulan dokumentasi proses digambarkan pada gambar berikut ini.



Gambar 7.5. Hirarki Proses

K. SERTIFIKASI PROSES

Saat ini, memperoleh sertifikasi atau peringkat dari badan standar atau pemilik model proses telah dianggap sangat penting, karena banyak organisasi yang mengalihdayakan pekerjaan pengembangan perangkat lunak bernilai tinggi menggunakan sertifikasi dan peringkat tersebut untuk membuat daftar calon vendor mereka. Oleh karena itu, buku tentang kualitas perangkat lunak sebagai jaminan tidak dapat lengkap tanpa menyentuh subjek ini. Saat ini, ada dua model populer: ISO 9000 dan CMMI®.

Sertifikat kepatuhan dengan seri standar ISO 9000 diberikan oleh auditor utama yang berwenang, yang melakukan audit pada sampel proyek, menilai kepatuhan manajemen proyek dan proses rekayasa perangkat lunak, dan mengaudit semua kelompok pendukung lainnya. Auditor menunjukkan ketidaksesuaian di mana praktik tidak sesuai dengan standar ISO 9000 yang berlaku. Auditor utama yang berwenang memberikan sertifikat kepatuhan jika tidak ada ketidaksesuaian atau jika ketidaksesuaian tidak signifikan. Jika ketidaksesuaian signifikan, auditor menolak untuk memberikan sertifikat atau menahan sertifikat sampai ketidaksesuaian diselesaikan secara memuaskan oleh organisasi. Seri standar ISO 9000 telah mengalami satu kali revisi sejak rilis aslinya pada tahun 1994. Salinan standar tersedia untuk dibeli dari ISO.

Peringkat kematangan kemampuan CMMI® diberikan oleh Software Engineering Institute (SEI) dari Carnegie Mellon University setelah menerima rekomendasi dari penilai utama yang berwenang. SEI merevisi metodologi untuk peringkat maturitas satu kali, dan metodologi ini disebut Metode Penilaian CMMI Standar untuk Peningkatan Proses (SCAMPI®).

Penilai pemimpin resmi melakukan penilaian SCAMPI® dengan tim yang terdiri dari orang-orang yang memenuhi syarat berdasarkan norma SCAMPI® dan dilatih oleh penilai pemimpin resmi. Penilai menunjukkan peluang organisasi untuk perbaikan jika ada yang ditemukan selama penilaian. Tim penilai, bekerja di bawah bimbingan dan arahan pemimpin pemberi pujian yang berwenang, mengumpulkan bukti dari sampel dokumen proyek yang dipilih dan kemudian menguatkan bukti itu dengan mewawancarai personel organisasi yang terlibat dalam implementasi proses. Jika, menurut pendapat tim penilai dan penilai utama yang berwenang, implementasi proses ditemukan menunjukkan bahwa model CMMI® sebagian besar diterapkan, penilai dan tim penilai mengajukan rekomendasi kepada SEI untuk memberikan peringkat kedewasaan kepada organisasi. SEI kemudian meninjau penilaian yang dilakukan dan memberikan peringkat maturitas yang direkomendasikan oleh tim penilai dan penilai.

Dokumen model CMMI® telah direvisi dua kali sejak rilis aslinya. Versi terbaru dari dokumen definisi model CMMI® dan Dokumen metodologi penilaian tersedia gratis bagi mereka yang tertarik untuk mempelajari model dan proses penilaian secara detail.

Kedua metodologi ini didasarkan pada cara kerja yang digerakkan oleh proses organisasi. Meskipun metode penilaian dan penghargaan akhir berbeda, langkah-langkah yang diikuti untuk mencapai penghargaan serupa:

1. Tentukan prosesnya.
2. Melaksanakan proses.
3. Memperbaiki proses berdasarkan umpan balik yang diperoleh dari lapangan tentang pelaksanaan proses.
4. Memantapkan proses, yang meliputi internalisasi dan kelembagaan alisasi kerja yang digerakkan oleh proses.

ISO memberikan sertifikat kepatuhan dan melakukan audit pengawasan sekali setiap enam bulan untuk memastikan kepatuhan yang berkelanjutan. SEI memberikan peringkat maturitas kemampuan; level 2 adalah peringkat minimum yang diberikan dan level 5 maksimum. Dalam adaptasi CMMI®, sebuah organisasi dimulai di level 2 dan lulus ke level 5 selama periode waktu tertentu dan penilaian yang berurutan. Peringkat jatuh tempo berlaku untuk jangka waktu tiga tahun. Untuk organisasi pengembangan perangkat lunak, CMMI® lebih sering diperlukan di Amerika Serikat dan sertifikat ISO 9000 diterima di sebagian besar belahan dunia lainnya.

L. RINGKASAN

Proses adalah definisi keseluruhan dari aktivitas organisasi utama. Ini adalah sebuah jaringan prosedur yang saling terkait untuk melakukan suatu aktivitas dan tingkat atas dokumen di mana dokumen lain memberikan rincian kegiatan. Stabilisasi proses diperlukan bagi organisasi untuk menghasilkan produk yang dapat diprediksi hasil. Namun, bukan berarti perbaikan proses tidak diperlukan. Hal ini hanya berarti bahwa suatu proses harus, pada umumnya, stabil. Perbaikan dipengaruhi berdasarkan pemicu, dan diterapkan untuk meningkatkan kualitas atau produktivitas atau untuk menyederhanakan pekerjaan. Stabilisasi proses menjadi mungkin, hanya setelah proses telah didefinisikan.

M. SOAL LATIHAN

1) Sebutkan komponen dari proses!

2) Gambarkan hirarki proses dan berikan pernjelasan!

3) Sebutkan proses pengembangan perangkat lunak!Jelaskan !

4) Gambarkan stabilisasi proses!Jelaskan!

5) Sebutkan 3 langkah untuk mencapai kualitas proses!

BAB VIII
PARADIGMA BARU UNTUK KUALITAS PERANGKAT LUNAK

A. PARADIGMA SERTIFIKASI SAAT INI

Kapan pun ada kelonggaran bagi penyedia untuk mengambil jalan pintas, pemerintah turun tangan untuk merumuskan peraturan dan checks and balances untuk melindungi calon korban jika asosiasi profesional atau badan industri gagal melakukannya. Misalnya, jika Anda ingin memulai sebuah perusahaan, perusahaan tersebut harus terdaftar di instansi pemerintah, dan Anda harus menyerahkan laporan berkala. Jika perusahaan Anda meminjam uang, maka Anda harus secara berkala menyampaikan laporan posisi keuangannya kepada pemerintah dan juga kepada pemberi pinjaman. Jika perusahaan Anda mengumpulkan uang melalui penawaran umum, ada sejumlah lembaga yang akan mengaturnya. Sebuah perusahaan publik membutuhkan departemen internal hanya untuk mematuhi peraturan tersebut. Mengapa? Karena perusahaan yang menerima uang dari masyarakat bertanggung jawab kepada publik melalui jasa baik pemerintah.

Perusahaan publik harus diaudit oleh firma audit yang terakreditasi untuk memastikan bahwa akunnya teratur meskipun perusahaan memiliki pejabat internal yang berkualifikasi yang menyiapkan akun dan auditor internal yang sama kompetennya. yang mengaudit akun tersebut. Proses ini hampir sama dengan secara berkala meminta mobil Anda disertifikasi laik jalan oleh lembaga terakreditasi, bahkan jika Anda seorang insinyur otomotif dan tahu bahwa mobil Anda layak untuk dikendarai. Auditor eksternal pertama-tama memeriksa keberadaan dan ketekunan pengendalian internal di perusahaan dan kemudian mengaudit sampel acak artefak keuangan. Jika auditor eksternal menemukan bahwa pengendalian internal tidak memuaskan, audit dihentikan dan auditor melaporkan perusahaan kepada otoritas pemerintah yang sesuai. Kegiatan ini dilakukan tanpa gagal, karena bagaimana perusahaan menggunakan uangnya (dan uang publik) mempengaruhi publik, dan pemerintah berusaha melindungi publik dari potensi penipuan.

Kualitas produk juga mempengaruhi masyarakat. Asosiasi konsumen, forum berpakaian ulang, undang-undang kriminal, dll. ada untuk melindungi masyarakat dari produk berkualitas rendah. Ketika sebuah bangunan dibangun, maka harus disertifikasi sebagai aman sebelum orang dapat menempatinya. Industri konstruksi diatur oleh inspeksi dan sertifikasi. Lembaga sertifikasi memeriksa catatan konstruksi serta struktur aktual untuk memastikan bahwa kualitas bahan yang tepat dan metode yang tepat digunakan. Setelah lembaga sertifikasi puas dengan temuannya, bangunan disertifikasi layak untuk ditempati.

Di masa lalu, produsen mengembangkan kontrol internal untuk kualitas melalui departemen jaminan kualitas (QA). Industri manufaktur dibentuk sebagai asosiasi, seperti National Electrical Manufacturers Association, Inter National Telecommunications Union, Solar Energy Industries Association, the Biotechnology Industry Organization, dll., untuk mengembangkan sejumlah standar kualitas dan praktik manufaktur yang baik untuk memastikan kualitas dalam produk. Organisasi di sektor jasa juga membentuk asosiasi untuk mendefinisikan mereka sendiristandar kualitas (Asosiasi Hotel dan Asosiasi Agen Perjalanan,

misalnya) dan untuk memastikan kepatuhan terhadap standar tersebut. Bahkan media memiliki dewan sendiri untuk mengawasi perilaku anggotanya.

Singkatnya, ada pengakuan sukarela di sektor manufaktur, jasa, dan konstruksi bahwa kualitas merupakan aspek penting dari produk apa pun, dan upaya siap dilakukan untuk terus meningkatkan kualitas produk atau layanan dengan membentuk pengendalian internal. Tidaklah berlebihan untuk mengatakan bahwa organisasi manufaktur atau jasa yang bereputasi baik yang tidak memiliki departemen kualitas jarang terjadi.

Pada tahun 1994, Organisasi Internasional untuk Standardisasi (ISO) merilis seri standar ISO 9000, yang berhubungan dengan kualitas proses. Ada banyak bidang manufaktur dan konstruksi di mana proses sangat penting, seperti menuangkan beton dan penempaan dan perlakuan panas logam. Ambil contoh, pembuatan bola lampu: vakum internal tidak dapat diuji setelah bola lampu disegel. Banyak proses seperti ini tidak dapat diperiksa atau diuji tanpa merusak komponen. Namun, tidak ada badan standar yang mengembangkan standar kualitas proses sampai perangkat lunak mulai memainkan peran yang semakin meningkat dalam kehidupan kita. Baru kemudian menjadi jelas bahwa kualitas perangkat lunak meninggalkan banyak hal yang diinginkan, dan kebutuhan akan standar kualitas proses diakui.

Tidaklah umum bagi perusahaan pengembangan perangkat lunak untuk memiliki departemen kualitas, dan karena alasan ini, kontrol internal untuk memastikan kualitas hasil, lebih sering tidak ada. Pentingnya fakta ini disorot oleh perangkat lunak peran penting sekarang bermain dalam kehidupan kita. Perangkat lunak mengontrol sebagian besar, jika tidak semua, komunikasi, perjalanan, senjata, distribusi daya, dan peralatan rumah tangga. Hal ini mengganggu untuk menyadari bahwa kualitas mengambil kursi belakang dalam organisasi yang mengembangkan perangkat lunak. Namun, sejauh ini tidak ada konsumen yang menuntut kerugian yang disebabkan oleh perangkat lunak dan diberikan kompensasi atas kerusakan.

Tidak ada peraturan atau lembaga pemerintah yang berperan sebagai anjing penjaga atas kualitas yang diberikan oleh organisasi pengembangan perangkat lunak. Mereka tidak mengirimkan laporan mengenai kualitas kepada siapa pun. Diktum "biarkan pembeli berhati-hati" paling tepat untuk produk perangkat lunak, karena pembeli tidak dapat menguji produk perangkat lunak secara komprehensif sebelum membelinya. Sebaliknya, misalnya, ketika Anda membeli mobil, Anda dapat mengujinya, dan Anda dapat membawa mobil kembali ke dealer dan meminta perbaikan jika tidak berfungsi selama masa garansi atau membawanya ke bengkel setelah masa garansi habis. kedaluwarsa. Tetapi dapatkah Anda membawa produk perangkat lunak kembali ke perusahaan yang mengembangkannya dan meminta perbaikan? Apakah ada yang namanya stasiun layanan perangkat lunak? Pengemudi dapat mengendarai mobil, yang harganya jauh lebih mahal daripada kebanyakan perangkat lunak, selama 15 menit dan mengetahui apakah merupakan kendaraan yang berkualitas. Bisakah pengguna menguji sistem operasi komputer dan benar-benar tahu dalam beberapa jam jika ada cacat? Dapatkah konsumen benar-benar memahami perbedaan antara komputer berbasis Windows, komputer berbasis Linux, dan Mac dan memutuskan mana yang paling sesuai dengan kebutuhannya?

Organisasi perangkat lunak bertujuan untuk mendapatkan sertifikat dari auditor utama untuk menunjukkan bahwa mereka mematuhi seri standar ISO 9000 atau untuk mendapatkan peringkat kematangan kemampuan level 2, 3, 4, atau 5 dari Integrasi Model Kematangan

Kemampuan (CMMI®) penilai memimpin. Namun pencapaian tersebut bukanlah jaminan kualitas perangkat lunak.

B. KESALAHAN SERTIFIKASI

Keadaan standardisasi dalam industri pengembangan perangkat lunak tidak sampai ke tingkat di bidang manufaktur. Benar, Institute of Electrical and Electronics Engineers (IEEE) merilis standar rekayasa perangkat lunak , tetapi standar ini adalah bukan standar kualitas perangkat lunak . Sebaliknya, mereka lebih merupakan pedoman daripada standar dalam arti kata yang ketat dan oleh karena itu terbuka untuk interpretasi dan adaptasi. Konsep manajemen kualitas total menunjukkan bahwa mengikuti proses yang ditentukan akan memastikan kualitas perangkat lunak, dan seri standar ISO 9000 dan Model Kematangan Kemampuan (CMM®) dan CMMI® dari Software Engineering Institute (SEI) sering digunakan oleh pengembangan perangkat lunak. organisasi untuk menanamkan kualitas dalam produk mereka. Namun, standar, pedoman, dan model ini semuanya memiliki keterbatasan dan kekurangan. Standar definitif yang dikembangkan oleh asosiasi industri, serupa dengan tingkat yang ada di industri manufaktur, konstruksi, dan jasa, tidak ada di industri pengembangan perangkat lunak.

Dengan dirilisnya standar dan pedoman di atas, banyak organisasi yang ingin mengalihdayakan pekerjaan pengembangan mereka mulai mendesak agar organisasi pengembangan perangkat lunak disertifikasi, terutama untuk CMMI® (meskipun sertifikasi sebenarnya adalah peringkat kematangan kemampuan). Memegang sertifikat membuka pintu untuk penawaran, dan kurangnya sertifikat menutup pintu itu. Organisasi pengembangan perangkat lunak mulai aktif mencari sertifikasi hanya untuk mempertahankan pangsa pasar mereka. Tak heran, organisasi sertifikasi mulai bermunculan bak jamur. Banyak organisasi pengembangan perangkat lunak menerima sertifikasi ISO 9000 yang didambakan dan dalam banyak kasus baik ISO maupun CMMI®.

Organisasi pengembang perangkat lunak juga dapat memperoleh peringkat kedewasaan dari model kedewasaan lainnya, seperti Model Kematangan Pengujian, Model Kematangan Orang, Model Kematangan Kemampuan Rekayasa Perangkat Lunak, Model Kematangan Kemampuan Layanan TI, dll. Lee Copeland mencantumkan 34 model kedewasaan dalam artikelnya " The Maturity Maturity Model™ (M3)" di situs Web StickyMinds.

Fokus asli dari seri standar ISO 9000 adalah pada kualitas, dengan "Sistem Manajemen Mutu" sebagai dokumen utamanya dan "Kebijakan Mutu" sebagai tulang punggung proses organisasi. Industri, bagaimanapun, melemahkan proses ini menjadi hanya proses operasi organisasi, sehingga mengurangi penekanan pada kualitas dan mengalihkan fokus dari kebijakan mutu organisasi ke visi organisasi dan dari sasaran mutu ke sasaran organisasi. CMMI® melangkah lebih jauh dengan menyatakan bahwa tujuan dari suatu proses harus mencapai tujuan organisasi. Kualitas kiriman jelas telah mengambil tempat di belakang.

C. KRITIK TERHADAP MODEL KEMATANGAN

Salah satu kritik utama dari model kedewasaan yang disebutkan dalam bab ini adalah bahwa mereka semua menekankan tujuan bisnis organisasi, bukan kualitas produk (atau hasil)! Keyakinan bahwa organisasi yang digerakkan oleh proses memberikan kualitas tidak pada tempatnya. Pertimbangkan isu-isu faktual ini: (1) proses itu sendiri mungkin cacat, (2) setiap proses memiliki celah, (3) pengembang lebih fokus untuk menyesuaikan diri dengan proses daripada mencapai keunggulan dalam kualitas, dan (4) manajemen lebih fokus pada penyampaian dan menjual dari pada kualitas.

Tugas kepala kualitas yang malang, jika ada, hanya berkoordinasi dengan lembaga sertifikasi, karena dia tidak memiliki kendali atas kualitas produk. Di banyak organisasi, orang yang memegang gelar "kepala kualitas" (seringkali di bawah sebutan lain, seperti kepala kelompok proses rekayasa perangkat lunak, koordinator kualitas, manajer kualitas, direktur kualitas, dll.) tidak benar-benar memenuhi syarat atau cukup berpengalaman. untuk memegang posting ini atau tidak memiliki pengetahuan yang cukup tentang konsep dan alat kualitas.

Model kedewasaan kurang fokus pada pengembangan perangkat lunak dan memastikan bahwa kualitas dibangun di dalamnya dan lebih pada proses dukungan. CMMI® untuk pengembangan (versi 1.2) memiliki lebih banyak area proses untuk manajemen proyek (perencanaan proyek, manajemen proyek terintegrasi, manajemen risiko, manajemen konfigurasi, pemantauan dan pengendalian proyek, manajemen proyek kuantitatif, manajemen perjanjian pemasok, dan manajemen persyaratan—total delapan) daripada yang memiliki area proses untuk kualitas (QA proses dan produk, validasi, dan verifikasi—total tiga). Ini hanya memiliki tiga area proses (integrasi produk, pengembangan persyaratan, dan solusi teknis) untuk pengembangan perangkat lunak. Hal ini hanya memiliki dua area proses (fokus proses organisasi dan pengembangan proses organisasi) untuk definisi proses organisasi dan hanya empat area proses (analisis dan resolusi kausal, analisis dan resolusi keputusan, pengukuran dan analisis, dan kinerja proses organisasi) untuk pengukuran dan analisis. Dua area proses yang tersisa adalah inovasi dan penyebaran organisasi dan pelatihan organisasi. Dengan demikian, fokus pada kualitas berkurang, karena hanya merupakan bagian dari 3 dari 22 area proses yang dibutuhkan organisasi pengembangan perangkat lunak untuk menerima peringkat.

Lebih jauh lagi, model kedewasaan tidak menuntut agar model tersebut benar-benar diimplementasikan. Mereka menerima "sebagian besar dilaksanakan" sebagai cukup untuk memberikan sertifikat atau peringkat. Model itu sendiri tidak didefinisikan secara ketat dan dibuat sangat fleksibel sehingga praktik yang diperlukan terbuka untuk interpretasi. Beberapa mengizinkan "praktik alternatif" sebagai pengganti praktik yang ditentukan dalam model. Tingkat fleksibilitas ini memungkinkan organisasi untuk melakukan apa yang mereka inginkan dan tetap disertifikasi sebagai sesuai dengan model.

Kritik kedua terhadap model kedewasaan saat ini adalah bahwa model tersebut tidak menentukan ambang batas kualitas apa pun untuk mencapai sertifikasi. Kesesuaian dengan proses yang ditentukan sendiri sudah memadai. Sebagai contoh, misalkan standar untuk alat listrik mendefinisikan resistansi isolasi dalam istilah kuantitatif (katakanlah satu megaohm) sehingga orang tidak tersengat listrik saat menangani alat tersebut. Namun tidak ada standar

rekayasa perangkat lunak yang mendefinisikan kepadatan cacat, misalnya, untuk aplikasi keuangan.

Kritik lain dari model proses atau model kematangan adalah bahwa mereka tidak menentukan berapa tahun organisasi harus beroperasi sebelum cukup matang untuk sertifikasi. Oleh karena itu, bahkan organisasi berusia satu tahun dapat memperoleh sertifikasi. Bahkan, organisasi satu orang dapat disertifikasi.

Kritik lain terhadap model kedewasaan, seperti disebutkan sebelumnya, adalah bahwa model tersebut tidak menentukan sasaran mutu apa pun yang harus dicapai untuk memperoleh sertifikasi. Kesesuaian belaka atau menunjukkan bukti kesesuaian hanya dalam enam proyek atau kurang sudah cukup bagi organisasi untuk disertifikasi, tanpa perlu menunjukkan bahwa organisasi tersebut telah mencapai kondisi kualitas yang dipersyaratkan.

Pemilik model kedewasaan tidak memantau kinerja aktual organisasi setelah mereka disertifikasi. ISO melakukan audit pengawasan setengah tahunan sepintas, tetapi CMMI® mengharuskan organisasi untuk dinilai kembali setiap tiga tahun. Pemilik model maturitas tidak mengetahui apakah kualitas suatu proses organisasi telah meningkat atau apakah jumlah keluhan pelanggan telah berkurang karena mereka tidak melacaknya.

D. PERTIMBANGAN KEUANGAN

Lembaga sertifikasi mengenakan biaya tinggi ($200 per jam adalah tarif rata-rata, dengan periode penilaian mulai dari dua hari hingga tiga minggu). Misalkan auditor atau penilai menolak sertifikasi salah satu kliennya. Menurut Anda bagaimana kemungkinan auditor atau penilai ini akan menerima banyak panggilan dari organisasi lain di masa depan? Langsing tidak ada. Oleh karena itu, yang terbaik yang dapat dilakukan penilai untuk mencegah bisnis penilaiannya tutup adalah dengan membatalkan tugas dengan organisasi jika dia tidak puas dengan persiapannya. Dengan kata lain, auditor atau penilai tidak mungkin mengambil risiko dicap terlalu ketat. Auditor atau penilai yang menawarkan sertifikasi dengan minimal rewel adalah yang paling dicari.

Sertifikat yang diberikan kepada auditor dan penilai yang memungkinkan mereka mengeluarkan sertifikat kepada organisasi terlalu mudah dibagikan. Norma tidak terlalu ketat, dan mereka tidak mengamanatkan pengalaman pengembangan perangkat lunak sama sekali. Tanpa pengalaman pengembangan perangkat lunak, bagaimana auditor atau penilai dapat menilai kemampuan organisasi pengembangan perangkat lunak?

Organisasi sertifikasi, seperti kebanyakan organisasi adalah bisnis nirlaba yang memiliki biaya yang harus dipenuhi, target yang harus dicapai, dan pertumbuhan yang harus dicapai. Organisasi-organisasi ini menghasilkan uang dengan menerbitkan sertifikat, dan itulah salah satu alasan mengapa begitu banyak sertifikat diterbitkan dan diterbitkan dengan mudah.

E. METODE PENILAIAN

Kritik dapat dengan mudah dibuat dari proses penilaian itu sendiri. Penilai mengevaluasi bukti yang disajikan kepada mereka, yang membuat metode penilaian lebih merupakan audit

kesesuaian daripada audit investigasi. Apa jaminan bahwa data tidak bias untuk memenuhi persyaratan penilai? Jika buku akuntansi (yang tunduk pada audit independen menurut undang-undang) dapat "dimasak", mengapa tidak data yang akan disajikan untuk sertifikasi? Dalam banyak kasus, organisasi penilai juga menjadi konsultan proses untuk sebuah organisasi.

F. RINGKASAN

Fokus asli dari seri standar ISO 9000 adalah pada kualitas, dengan “Sistem Manajemen Mutu” sebagai dokumen utamanya dan “Kebijakan Mutu” sebagai tulang punggung proses organisasi. Industri, bagaimanapun, melemahkan proses ini menjadi hanya proses operasi organisasi, sehingga mengurangi penekanan pada kualitas dan mengalihkan fokus dari kebijakan mutu organisasi ke visi organisasi dan dari sasaran mutu ke sasaran organisasi. CMMI® melangkah lebih jauh dengan menyatakan bahwa tujuan dari suatu proses harus mencapai tujuan organisasi. Kualitas kiriman jelas telah mengambil tempat di belakang.

G. SOAL LATIHAN

1) Apa yang menjadi fokus seri standar ISO 9000?
2) Sebutkan pertimbangan isu-isu faktual dari kritik terhadap model kematangan!
3) Apa yang dimaksud dengan Organisasi sertifikasi?
4) Bagaimana metode penilaian perangkat lunak?
5) Bagaimana Paradigma Sertifikasi Saat Ini?

DAFTAR PUSTAKA

Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Schmitt, P. H., & Ulbrich, M. (2016). *Deductive Software Verification – The KeY Book: From Theory to Practice* (1st Edition ed.). Cham, Switzerland: Springer.

Chemuturi, M. (2018). *Software Design: A Comprehensive Guide to Software Development Projects* (1st edition ed.). Florida: CRC Press.

Durivage, M. A., & Mehta, B. (. (2016). *Practical Process Validation.* Perth, Australia: Quality Press.

Griffor, E. (2016). *Handbook of System Safety and Security: Cyber Risk and Risk Management, Cyber Security, Threat Analysis, Functional Safety, Software Systems, and Cyber Physical Systems* (1st edition ed.). Massachusetts: Syngress.

Information Resources Management Association. (2021). *Research Anthology on Agile Software, Software Development, and Testing.* Pennsylvania: IGI Global.

Jones, E. (2014). *Quality Management for Organizations Using Lean Six Sigma Techniques* (1st edition ed.). Florida: CRC Press.

Ketola, J., & Roberts, K. (2001). *ISO 9000:2000 In a Nutshell* (2nd edition ed.). California: Paton Press.

Levin, M. A., Kalal, T. T., & Rodin, J. (2019). *Improving Product Reliability and Software Quality: Strategies, Tools, Process and Implementation (Quality and Reliability Engineering Series)* (2nd Edition ed.). New Jersey: John Wiley & Sons, Inc.

Luthra, S., Garg, D., Agarwal, A., & Mangla, S. K. (2020). *Total Quality Management (TQM): Principles, Methods, and Applications* (1st edition ed.). Florida: CRC Press.

Wagner, S. (2013). *Software Product Quality Control.* Berlin: Springer Science & Business Media.

Wiegers, K., & Beatty, J. (2013). *Software Requirements (Developer Best Practices)* (3rd Edition ed.). Washington, United States: Microsoft Press.

GLOSARIUM

CD
Compact disk

CEO
Chief executive officer

CMM®
Capability Maturity Model

CMMI®
Capability Maturity Model Integration

CNC
Computer numerical control

COTS
Commercial off-the-shelf

CPQR
Composite product quality rating

DBMS
Database management system

DIR
Defect injection rate

DRE
Defect removal efficiency

EQAR
Effectiveness of organizational quality assurance activities rating

ETR
Exhaustiveness of testing rating

FP
Function point

GUI
Graphical user interface

IDE
Interactive development environment

IEEE
Institute of Electrical and Electronics Engineers

ISO
International Organization for Standardization

LOC
Lines of code

MTBF
Mean time between failures

MTTR
Mean time to repair

NC
Nonconformance

NCR
Nonconformance report

OER
Organizational environment rating

PL
Project leader

PRCR
Peer review coverage rating

QA
Quality assurance

RAM
Random access memory

RDBMS
Relational database management system

SCAMPI®
Standard CMMI Appraisal Method for Process Improvement

SEI
Software Engineering Institute of Carnegie Mellon University

SPM
Software project manager

TBD

To be determined

TPM
Total productive maintenance

TQM
Total quality management

UAT
User acceptance testing

UTCR
Unit testing coverage rating

Y2K
Year 2000

INDEKS

TEKNIK MENJAMIN KUALITAS

BAGI PENGEMBANG PERANGKAT LUNAK

Profil Penulis

Sindhu Rakasiwi, S.Kom., M.Kom

Penulis lahir di kota Solo, namun sekarang ini tinggal di Kabupaten Semarang, merupakan dosen tetap di Universitas Sains dan Teknologi Semarang, dan juga menjabat sebagai Sekretaris Program Studi pada program studi Teknik Komputer. Riwayat pendidikan penulis adalah telah menyelesaikan pendidikan jenjang S1 dan S2 di Universitas Dian Nuswantoro (Udinus) Semarang, pada program studi Teknik Informatika. Karya penelitian telah banyak dihasilkan oleh penulis pada bidang ilmu TI baik di jurnal-jurnal nasional terakreditasi, maupun di jurnal internasional.

TEKNIK MENJAMIN KUALITAS

BAGI PENGEMBANG PERANGKAT LUNAK

Buku ajar teknik menjamin kualitas ini isinya mencakup seluruh uraian materi yang diperlukan bagi siapapun terutama bagi para mahasiswa dan pengembang perangkat lunak untuk menjamin kualitas atau mutu dari suatu perangkat lunak yang dikembangkan. Materi disajikan secara lengkap, mulai dari dasar pemahaman tentang penjaminan mutu, dimensi-dimensi dari kualitas, perangkat, veriikasi, validasi, tinjauan keandalan dan proses, hingga ke pandangan baru yang ada. Target penulisan buku ini tidak hanya sebagai referensi untuk kalangan mahasiswa, melainkan juga untuk umum. Buku ini dilengkapi dengan ringkasan materi serta soal-soal latihan pada bagian akhir di setiap bab.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-19-2 (PDF)
9 786238 120192