HALAMAN PENGESAHAN LAPORAN AKHIR

1.	Judul Diktat	: Manajemen Kualitas Perangkat Lunak
2.	Biodata	:
a.	Nama Lengkap	: Hero Yudo Martono, ST. MT
b.	Jenis Kelamin	: Laki-laki
c.	NIP	: 19781103.200501.1.002
d.	Jabatan Fungsional	: Lektor
e.	Jabatan Struktural	: Sekretaris Unit Penelitian
f.	Bidang Keahlian	: Sistem Informasi dan Perangkat Lunak
g.	Fakultas/Jurusan	: Teknik Informatika
h.	Perguruan Tinggi	: Politeknik Elektronika Negeri Surabaya

Surabaya, 1 Oktober 2011

Mengetahui,
Ketua Jurusan Teknik Informatika

Penulis

Arna Fariza, S.Kom, M.Kom
NIP. 19710807.199903.2.001

Hero Yudo Martono,ST. MT
NIP. 19781103.200503.1.002

Menyetujui
Ketua Lembaga

Eko Henfri Binugroho, S.ST. M.Sc
NIP. 19781103.200503.1.002

PRAKATA

Pertama-tama perkenankan kami memanjatkan puji syukur ke hadirat Tuhan Yang Maha Esa atas terbitnya Buku Diktat Kuliah "MANAJEMEN KUALITAS PERANGKAT LUNAK" pada jurusan Teknologi Informasi Politeknik Elektronika Negeri Surabaya (PENS). Buku ini diharapkan dapat dijadikan acuan dalam rangka proses belajar mengajar serta mampu memberikan kontribusi pada perkembangan ilmu perangkat lunak. Pembuatan buku ini merupakan salah satu bentuk ekspresi dari kewajiban seorang dosen sebagai hasil pengalaman mengajar mata kuliah tertentu dalam beberapa waktu. Buku ini diharapkan dapat dijadikan bahan diskusi seputar kegiatan pembuatan perangkat lunak yang mempertimbangkan faktor kualitas sebagai faktor yang penting.

Buku ini merupakan intisari yang penulis rangkum dari beberapa buku referensi mengenai kualitas perangkat lunak dan pengalaman dalam melakukan proses pembangunan perangkat lunak. Namun penulis sadar bahwa tidak gading yang tak retak, sehingga penulis menampung semua saran dan anjuran demi terciptanya diktat ini menjadi lebih baik.

Kami berharap buku diktat ini dapat bermanfaat sebagai acuan pembelajaran mata kuliah MKPL di lingkungan Jurusan Teknologi Informasi Politeknik Elektronika Negeri Surabaya, untuk kurang lebihnya kami ucapkan terima kasih yang sebesar-besarnya.

Surabaya, 1 Oktober 2011

DAFTAR ISI

Bab1. Tantangan Terhadap Kualitas Perangkat Lunak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Mengidentifikasi karakteristik unik pada sebuah perangkat lunak baik sebagai sebuah produk maupun sebagai proses produksi yang membenarkan perlakuan yang terpisah terhadap permasalahan kualitas
2. Mengenali karakteristik lingkungan dimana pengembang dan pemelihara perangkat lunak profesional berada
3. Menjelaskan kesulitan utama yang dihadapi oleh tim pengembang dan pemelihara perangkat lunak di lingkungan tempat mereka bekerja.

1.1. Keunikan tentang Jaminan Kualitas Perangkat Lunak

Perbedaan karakteristik antara produk perangkat lunak dan produk industri adalah dalam hal :

1) Kerumitan produk
2) Kenampakan produk
3) Proses pengembangan dan pembuatan produk

   Dalam memproduksi sebuah produk, kita dapat mendeteksi kegagalan dalam memproses sebuah produk dalam beberapa tahap dibawah :

   a) Pengembangan produk
   b) Perencanaan pembuatan produk
   c) Perakitan

   Sebagai perbandingan terhadap produk industri, maka produk perangkat lunak tidak beruntung dari sisi pendeteksian cacat produk pada tiga fase proses produksi. Fase dimana terdapat kemungkinan mampu mendeteksi cacat adalah fase pengembangan. Mari kita perikasa masing-masing fase untuk melihat kontribusi dalam hal pendeteksian cacat :

   a) Pengembangan produk
   b) Perencanaan pembuatan produk
   c) Perakitan

Table 1.1 Faktor pendeteksi kegagalan dalam produk perangkat lunak vs produk industri

Karakteristik	Produk Perangkat Lunak	Produk Industri
Kerumitan	Biasanya produk yang sangat rumit mempunyai sangat banyak pilihan operasional	Tingkat kerumitan lebih rendah, mempunyai sedikitnya beberapa ribu pilihan operasi
Kenampakan produk	Produk tidak nampak (invisible), sulit mendeteksi cacat atau kesalahan dengan melihat disket atau cd software	Produk terlihat (visible), memperkenankan pendeteksian yang efektif terhadap cacat atau kesalahan dengan melihat
Proses alami dari pengembangan dan produksi	Kesempatan untuk mendeteksi kesalahan hanya muncul hanya pada satu tahap yaitu pengembangan produk	Kesempatan untuk mendeteksi kesalahan muncul pada semua tahap pengembangan dan produksi

1.2. Lingkungan Metode SQA dikembangkan

Pengembangan perangkat lunak oleh beberapa individu dan dalam beragam situasi meliputi beberapa kebutuhan yaitu :

- Siswa dan mahasiswa mengembangkan perangkat lunak sebagai bagian dari tugas pendidikan.
- Amatir perangkat lunak mengembangkan perangkat lunak sebagai bagian hobi
- Profesional dalam teknik, ekonomi, manajemen dan bidang lain mengembangkan produk perangkat lunak untuk membantu mereka dalam pekerjaannya seperti melakukan perhitungan, ringkasan penelitian, kegiatan penelitian dan lain sebagainya.
- Pengembang perangkat lunak profesional (sistem analis dan programmer) mengembangkan produk perangkat lunak sebagai seorang profesional karir yang berperan sebagai karyawan dalam sebuah software house dengan cara mengembangkan perangkat lunak dan memelihara perangkat lunak pada industri kecil, industri keuangan maupun organisasi lain.

Mari kita mulai dengan memeriksa lingkungan pengembangan dan pemeliharaan perangkat lunak yang profesional yang merupakan pertimbangan utama dalam pengembangan metodologi SQA dan penerapannya. Karakteristik utama dari lingkungan ini sebagai berikut :

1) Kondisi perjanjian/kontrak
    - Daftar pendefinisian dari kebutuhan fungsional dari perangkat lunak yang akan dikembangkan atau diperlihara telah terpenuhi.

- Biaya proyek
- Jadwal proyek

2) Hubungan pelanggan dengan penyedia
3) Tim kerja yang dibutuhkan

Tiga faktor yang biasanya melatarbelakangi pembentukan sebuah tim proyek daripada menetapkan satu orang profesional adalah sebagai berikut :

- Kebutuhan akan penjadwalan waktu
- Kebutuhan akan keahlian yang berbeda untuk menyelesaikan proyek
- Keinginan untuk mendapatkan keuntungan dari profesional seiring dengan dukungan dan review yang akan meningkatkan kualitas proyek

4) Kerjasama dan koordinasi dengan tim perangkat lunak lain

Penyelesaian proyek, khususnya proyek untuk skala besar oleh lebih dari satu tim adalah kejadian yang umum dalam industri perangkat lunak, dalam banyak kasus kerjasama diperlukan dengan :

- Tim pengembang perangkat lunak dan perangkat keras lain dalam organisasi yang sama
- Tim pengembang perangkat lunak dan perangkat keras organisasi penyedia lain
- Tim pengembang perangkat lunak dan perangkat keras pelanggan yang merupakan bagian dari tim proyek.

5) Interface dengan sistem perangkat lunak lain.

Beberapa hal berikut dapat mengindentifikasi tipe utama dari interface :

- Interface inputan, dimana sistem perangkat lunak lain mengirimkan data sistem perangkat lunak kita
- Interface output, dimana sistem perangkat lunak kita mengirimkan data ke sistem perangkat lunak lain
- Interface input dan output ke papan kontrol mesin, seperti dalam sistem kontrol kesehatan dan laboratorium, peralatan pemrosesan baja.

1.3. Review Pertanyaan

1) Tiga faktor utama perbedaan produk perangkat lunak dengan produk industri
   - Identifikasi dan gambarkan perbedaan tersebut.

- Diskusikan cara perbedaaan ini berpengaruh terhadap SQA

2) Dinyatakan bahwa tidak ada kegiatan SQA yang penting untuk diterapkan dalam fase perancangan produksi untuk produk software.
   - Diskusikan pernyataan tersebut
   - Bandingkan kebutuhan perancangan produksi untuk model mobil baru dengan fase dari perancangan produksi yang diperlukan untuk mengeluarkan produk software baru.
3) Tujuh isu pokok dalam lingkungan pengembangan dan pemeliharaan perangkat lunak secara profesional :
   - Identifikasi dan gambarkan karakteristik ini
   - Lingkungan karakteristik mana yang berpengaruh terhadap usaha yang diperlukan untuk menyelesaikan pengembangan perangkat lunak dan proyek pemeliharaan ? sebutkan lima karakteristik dan jelaskan mengapa seorang profesional diperlukan.

Bab2. Apa Maksud Kualitas Perangkat Lunak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Mendefinisikan software, kualitas software dan jaminan kualitas software (SQA)
2. Membedakan software error (salah), faults(keliru), failure (gagal)
3. Mengidentifikasi ragam penyebab software error.
4. Menjelaskan tujuan dari kegiatan SQA
5. Membedakan dan menjelaskan perbadaan antara jaminan kualitas software (SQA) dan pengawasan kualitas
6. Menjelaskan hubungan antara SQA dan rekayasa software (software engineering)

2.1. Apa yang dimaksud dengan Software ?

Berdasarkan intuisi, ketika kita berpikir tentang perangkat lunak, kita membayangkan kumpulan dari instruksi dan pernyataan bahasa program yang secara bersama-sama membentuk sebuah program atau paket perangkat lunak. Program atau perangkat lunak ini biasanya mengacu pada sebuah kode. Cukup untuk memperhatikan kode untuk memastikan kualitas dari layanan yang disediakan oleh program perangkat lunak.

Definisi : Perangkat Lunak
Komputer program, prosedur dan dokumentasi yang mungkin dan data yang terkait dengan operasional dari sebuah sistem komputer.

Berdasarkan definisi IEEE tentang perangkat lunak dimana hampir sama dengan definisi dari ISO, maka daftar dari empat komponen dari perangkat lunak adalah sebagai berikut :

- Komputer program
- Prosedur
- Dokumentasi
- Data yang diperlukan untuk operasional sistem perangkat lunak.

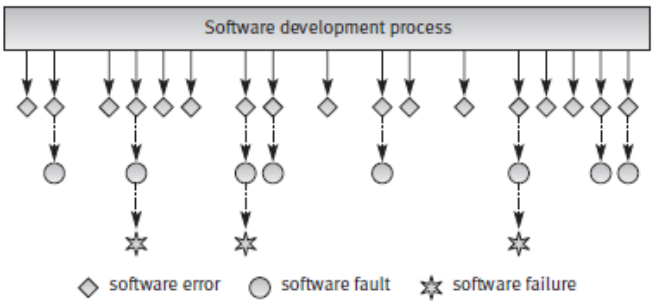
Semua empat komponen yang diperlukan untuk memastikan kualitas dari proses pengembangan perangkat lunak dan layanan pemeliharaan untuk beberapa alasan sebagai berikut :

- Komputer program, diperlukan karena kode akan mengaktifkan komputer untuk melakukan aplikasi yang diperlukan.
- Prosedur diperlukan, untuk mendefinisikan pesanan dan jadwal dimana program dilaksanakan, metode dilakukan dan orang yang bertanggung-jawab untuk melakukan kegiatan yang diperlukan untuk menerapkan perangkat lunak.
- Berbagai macam tipe dokumen diperlukan oleh pengembang, pengguna dan personel pemelihara. Dokumentasi pengembangan (laporan kebutuhan, laporan rancangan, deskripsi program) memperkenankan kerjasama dan koordinasi diantara anggota tim pengembang dan pemeriksaan yang efisien. Dokumentasi user memberikan deskripsi tentang ketersediaan aplikasi dan metode yang sesuai untuk mereka gunakan. Dokumentasi pemeliharaan menyediakan tim pemeliharaan semua informasi tentang kode, struktur dan tugas dari masing-masing modul. Informasi ini digunakan ketika mencari penyebab kesalahan atau merubah bahkan menambah sesuatu pada perangkat lunak yang telah ada.
- Data termasuk parameter, kode dan daftar nama yang diperlukan oleh user khusus untuk mengoperasikan perangkat lunak. Tipe lain yang penting dari data adalah data tes standar, untuk mengetahui apakan diperlukan perubahan kode atau data perangkat lunak dan bentuk kesalahan apa terjadi di perangkat lunak.

2.2. Klasifikasi penyebab software eror

Ada beberapa ungkapan pernyataan sebagai berikut :

- Kami telah menggunakan perangkat lunak Simplex HR dalam departemen Sumber Daya Manusia kami, kurang lebih tiga tahun dan kami tidak pernah mengalami kerusakan.
- Kami mulai menggunakan Simplex HR dua bulan yang lalu, kami memiliki banyak kesalahan sehingga kami berpikiran untuk mengganti paket perangkat lunak kami.
- Kami telah menggunakan paket perangkat lunak yang sama untuk hampir lebih empat tahun. Kami sangat puas sampai akhirnya bulan kemarin, ketika tiba-tiba kami mengalami beberapa kesalahan. Support Centre dari perangkat lunak ini mengklaim bahwa mereka tidak pernah mengalami kegagalan seperti yang kami alami meskipun meraka telah melayani sekitar 700 pelanggan yang menggunakan simple HR.



Gambar 2.1. Perangkat lunak eror, fauld dan failure

Gambar 2.1 menggambarkan hubungan antara perangkat lunak eror, fault dan failures. Dalam gambar ini proses pengembangan perangkat lunak menghasilkan 17 kesalahan perangkat lunak , hanya delapan diantaranya menyebabkan perangkat lunak fault, dan diantara yang fault ini hanya tiga yang menyebabkan perangkat lunak failure.

2.3. Definisi Kualitas Software

Kegagalan perangkat lunak disebabkan oleh buruknya kualitas dari perangkat lunak. Sangat penting untuk menyelidiki penyebab dari kesalahan dan mencegah timbulnya kesalahan tersebut. Sebuah perangkat lunak eror bisa karena “kode eror” , sebuah prosedur eror, dokumentasi eror atau data perangkat lunak eror. Hal ini seharusnya menegaskan bahwa penyebab dari semua ini adalah kesalahan manusia, yang dibuat oleh sistem analis, programmer, tester perangkat lunak, tim ahli dokumentasi, manajer dan kadang-kadang user dan perwakilan mereka. Bahkan di kasus yang jarang terjadi dimana kesalahan perangkat lunak mungkin disebabkan oleh lingkungan pengembangan. Hal ini juga beralasan bahwa untuk mengklaim bahwa ini kesalahan manusia yang disebabkan oleh kegagalan dari tool yang digunakan untuk pengembangan. Penyebab dari kegagalan perangkat lunak lebih jauh diklasifikasikan sebagai tahapan proses pengembangan perangkat lunak.

1) Kesalahan dalam mendefinisikan kebutuhan
2) Kesalahan komunikasi antara client dan pengembang
3) Timbulnya deviasi dari kebutuhan perangkat lunak
4) Kesalahan rancangan secara logical
5) Kesalahan koding
6) Kurangnya pemenuhan terhadap dokumentasi dan instruksi koding
7) Proses testing yang singkat

8) Kesalahan prosedur
9) Dokumentasi yang salah

2.4. Definisi dan tujuan Jaminan Kualitas Software

Pendahuluan tentang komponen perangkat lunak, kesalahan dan penyebabnya serta pengetahuan dimana eror merugikan kualitas dari perangkat lunak, telah menyiapkan kami untuk mendefinisikan tentang target kami yaitu : Perangkat Lunak Berkualitas.

Berdasarkan Definisi dari IEEE

Perangkat Lunak Berkualitas adalah

1. Derajat kondisi dimana sebuah sistem, komponen atau proses berhasil memenuhi kebutuhan yang khusus.
2. Derajat kondisi dimana sebuah sistem, komponen atau proses berhasil memenuhi kebutuhan user dan harapan user.

Berdasarkan Definisi dari Pressman's

Perangkat Lunak Berkualitas adalah

1. Pemenuhan secara eksplisit pernyataan terhadap fungsional dan kebutuhan performansi, standar dokumen pengembangan, karakteristik yang diharapkan oleh semua pengembang perangkat lunak professional.

Definisi dari Pressman's menyarankan tiga syarat untuk menjamin kualitas yang harus dipenuhi oleh pengembang sebagai berikut :

- Kebutuhan fungsional yang khusus, dimana utamanya mengacu pada keluaran dari sistem perangkat lunak.
- Standar kualitas perangkat lunak yang disebutkan di kontrak.
- Menggunakan praktis teknik perangkat lunak, yang mencerminkan praktis professional bidang perangkat lunak terkini, yang harus dipenuhi oleh pengembang meskipun tidak secara eksplisit disebutkan dalam kontrak.

2.5. Software engineering dan Jaminan Kualitas Software

Dalam bab ini kita akan membahas tentang :

- Alternatif definisi dari SQA
- Jaminan kualitas perangkat lunak dibandingkan dengan pengawasan kualitas perangkat lunak
- Tujuan dari SQA

2.5.1. Definisi Jaminan Kualitas Perangkat Lunak

Jaminan Kualitas Perangkat Lunak – Definisi IEEE
1. Semua tindakan yang diperlukan terencana dan sistematik untuk menyediakan kepercayaan diri yang cukup bahwa item atau produk yang dihasilkan mampu memenuhi kebutuhan teknikal.
2. Rangkaian kegiatan yang dirancang untuk mengevaluasi proses dimana produk dikembangkan atau dirangkai.

Definisi ini digolongkan sebagai berikut :

- Perencanaan dan penerapan yang sistematis. SQA berdasarkan pada perencanaan dan aplikasi dari tindakan yang beragam yang terintegrasi menjadi semua tahapan dalam proses pengembangan perangkat lunak.
- Mengacu pada proses pengembangan perangkat lunak
- Mengacu pada spesifikasi kebutuhan teknikal

2.5.2. Penjaminan Kualitas Perangkat Lunak vs Pengawasan Kualitas Perangkat Lunak

Dua istilah diatas menyajikan perbedaan konsep yaitu :

- Pengawasan Kualitas didefinisikan sebagai rangkaian kegiatan yang dirancang untuk mengevaluasi kualitas dari produk yang dikembangkan atau dirangkai, dengan kata lain, kegiatan yang mempunyai tujuan menahan produk-produk yang tidak berkualitas. Yang tepat adalah pemeriksaan pengendalian mutu dan kegiatan lainnya berlangsung pada proses pengembangan atau perakitan produk harus selesai sebelum barang dikirimkan ke pelanggan.
- Tujuan utama dari penjaminan kualitas adalah untuk meminimalkan biaya dengan cara menjamin kualitas melalui berbagai macam kegiatan yang dilakukan selama tahap pengembangan dan perakitan. Kegiatan ini mencegah penyebab timbulnya eror dan mendeteksi serta membenahinya sejak awal proses pengembangan.

2.5.3. Tujuan Kegiatan SQA

Tujuan dari kegiatan SQA (Proses Pengembangan) adalah :

1) Memastikan level penerimaan kepercayaan diri bahwa perangkat lunak mampu memenuhi kebutuhan fungsional secara teknikal.
2) Memastikan level penerimaan kepercayaan diri bahwa perangkat lunak akan mampu memenuhi syarat waktu/jadwal dan biaya
3) Kegiatan pendahuluan untuk peningkatan dan efisiensi yang lebih besar terhadap kegiatan SQA dan pengembangan perangkat lunak. Hal ini berarti bahwa peningkatan fungsional dan kebutuhan pengelolaan akan mampu dipenuhi dan juga mampu mengurangi biaya pengembangan perangkat lunak.

Tujuan dari kegiatan SQA (Pemeliharaan Perangkat Lunak) adalah :

1) Memastikan level penerimaan kepercayaan diri bahwa kegiatan pemeliharaan perangkat lunak akan memenuhi kebutuhan fungsional secara teknikal.
2) Memastikan level penerimaan kepercayaan diri bahwa kegiatan pemeliharaan perangkat lunak akan memenuhi kebutuhan dari sisi penjadwalan dan biaya
3) Kegiatan pendahuluan untuk peningkatan dan efisiensi yang lebih besar terhadap kegiatan SQA dan pengembangan perangkat lunak. Hal ini berarti bahwa peningkatan fungsional dan kebutuhan pengelolaan akan mampu dipenuhi dan juga mampu mengurangi biaya

2.6. Ringkasan

1. Sebutkan definisi dari perangkat lunak, kualitas perangkat lunak dan jaminan kualitas perangkat lunak.
2. Sebutkan perbedaan antara perangkat lunak eror, faults dan failures.
3. Sebutkan ragam penyebab perangkat lunak eror.
4. Jelaskan tujuan dari kegiatan jaminan kualitas perangkat lunak.
5. Bedakan dan jelaskan perbedaan antara jaminan kualitas perangkat lunak dan pengawasan kualitas.
6. Jelaskan hubungan antara jaminan kualitas perangkat lunak dengan rekayasa perangkat lunak.

Bab3. Faktor Kualitas Perangkat Lunak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan kebutuhan terhadap dokumen persyaratan secara menyeluruh dan karekteristik dari masing-masing dokumen tersebut.
2. Menjelaskan struktur (faktor dan kategori) dari model klasik McCall's
3. Mendata faktor selain dalam model McCall's yang disarankan sebagai alternatif model SQA.
4. Mengidentifikasi siapa yang tertarik dalam mendefinisikan tentang kebutuhan kualitas

3.1. Kebutuhan terhadap syarat Kualitas Perangkat Lunak

Beberapa kutipan pernyataan tentang kualitas perangkat lunak

- "Sistem informasi penjualan baru kami tampaknya sudah benar, faktur sudah benar, catatan persediaan sudah benar, diskon yang diberikan kepada klien kami persis mengikuti kebijakan diskon kita yang sangat rumit, tapi sistem informasi penjualan baru kita sering gagal, biasanya paling tidak dua kali sehari, setiap kali selama dua puluh menit atau lebih. Kemarin masalah terebut menghabiskan waktu satu jam setengah sebelum akhirnya terselesaikan dan kita bisa kembali bekerja. Bayangkan betapa memalukannya hal ini untuk disampaikan kepada manajer. Softbest, perusahaan perangkat lunak yang mengembangkan sistem penjualan terkomputerisasi kami, mengklaim tidak bertanggung jawab. . . . "
- "Baru setengah tahun yang lalu kami meluncurkan produk baru kami – detektor radar. Firmware RD-8.1, tertanam dalam produk ini, tampaknya akan menjadi faktor keberhasilannya. Tapi, ketika kami mulai merencanakan untuk mengembangkannya ke Eropa untuk versi produk, kami menemukan bahwa meskipun produk akan hampir sama, perangkat lunak departemen pengembangan kita perlu mengembangkan firmware baru, hampir semua desain dan program akan menjadi baru ".
- "Percaya atau tidak, 'Blackboard' paket perangkat lunak kami untuk para guru sekolah, diluncurkan hanya tiga bulan yang lalu, sudah terpasang di 187 sekolah. Tim pengembangan baru saja kembali dari liburan seminggu di Hawaii, bonus liburan mereka. Tapi tiba-tiba kita menerima keluhan setiap hari dari tim pemeliharaan 'Blackboard'. Mereka mengklaim bahwa kurangnya failuredetection fitur dalam perangkat lunak, selain programmer kurang akan manual book, menyebabkan mereka untuk berinvestasi lebih dari perkiraan waktu sebelumnya untuk menangani bug atau menambahkan

beberapa perubahan perangkat lunak kecil yang disepakati sebagai bagian dari pembelian kontrak dengan klien. "

- "Versi baru dari perangkat lunak kontrak peminjaman kami benar-benar akurat. Kami telah menerima 1200 permintaan pelanggan, dan memeriksa setiap output kontrak. Tidak ada kesalahan. Tapi kami menghadapi masalah yang tak terduga – pelatihan anggota staf baru untuk menggunakan software ini membutuhkan waktu sekitar dua minggu. Ini adalah masalah nyata dalam departemen pelanggan yang menerima penderitaan terbesar dari perputaran karyawan tersebut. . . . Tim proyek mengatakan bahwa mereka tidak memiliki cukup waktu untuk melakukan pelatihan sesuai jadwal yang ditentukan, tambahan dua sampai tiga bulan masa kerja dalam pelatihan akan diperlukan untuk memecahkan masalah tersebut. "Ada beberapa karakteristik umum untuk mengatasi semua "akan tetapi," itu:

Terdapat karakteristik yang hampir sama terhadap semua ini tapi :

- Semua proyek perangkat lunak dianggap memuaskan bila memenuhi persyaratan dasar untuk perhitungan yang benar(angka persediaan yang benar, rata-rata kelas's skor yang benar, bunga pinjaman yang benar, dll).
- Semua proyek perangkat lunak mengalami kegagalan yang berawal dari kinerja buruk di bidang-bidang penting seperti pemeliharaan, kehandalan, penggunaan kembali perangkat lunak, atau pelatihan.
- Penyebab kinerja yang buruk dari proyek perangkat lunak yang dikembangkan di daerah-daerah adalah kurangnya persyaratan yang telah ditetapkan untuk menutupi penting aspek fungsionalitas perangkat lunak.

Kebutuhan akan definisi yang komprehensif dari persyaratan.Ada beberapa kebutuhan dalam medefinisikan suatu persyaratan yang akan mencakup semua atribut dari perangkat lunak dan aspek penggunaan perangkat lunak, termasuk aspek kegunaan, aspek usabilitas, aspek pemeliharaan, dan sebagainya dalam menjamin kepuasan dari pelanggan.

Berbagai persoalan besar yang terkait dengan berbagai atribut perangkat lunak serta penggunaan dan pemeliharaan, sebagaimana didefinisikan dalam dokumen persyaratan perangkat lunak, dapat diklasifikasikan ke dalam kelompok konten yang disebut faktor kualitas(quality factors). Kami

mengharapkan tanggung jawab tim untuk menentukan kebutuhan sistem perangkat lunak untuk memeriksa kebutuhan penentuan persyaratan yang dimiliki masing masing faktor. Dokumen kebutuhan perangkat lunak yang diharapkan dapat berbeda dalam penekanan penempatannya pada berbagai faktor, dimana merupakan gambaran dari perbedaan yang ditemukan pada proyek perangkat lunak. Dengan demikian, kita dapat berharap bahwa tidak semua faktor akan universal "mewakili" dalam semua dokumen persyaratan. Kesepakatan berikutnya dengan klasifikasi persyaratan mutu faktor-faktor kualitas. Jelas, hanya berupa pendekatan untuk menutup topik ini.

3.2. Klasifikasi kebutuhan software menjadi faktor kualitas Software

Beberapa model faktor kualitas perangkat lunak dan kategorisasi mereka dalam kategori faktor telah diusulkan selama bertahun-tahun. Model klasik dari perangkat lunak faktor kualitas, disarankan oleh McCall, terdiri dari 11 faktor (McCall et al., 1977). Model berikutnya, terdiri dari 12 sampai 15 faktor, yang disarankan oleh Deutsch dan Willis (1988) dan oleh Evans dan Marciniak (1987). Alternatif model yang tidak berbeda jauh dari model McCall's. The McCall model faktor, meskipun seperempat abad waktu yang dibutuhkan untuk "pematangan"-nya, terus berlanjut untuk memberikan praktis yang up to-date pada metode untuk mengklasifikasikan kebutuhan perangkat lunak (Pressman, 2000).

Model Faktor MacCall’s mengklasifikasikan semua kebutuhan software menjadi 11 faktor kualitas software. Ke 11 faktor tersebut kemudian dikelompokkan menjadi 3 kategori sebagai berikut :

- Faktor operasional produk : correctness, reliability, efficiency, integrity, usability
- Faktor perbaikan produk : maintainability, flexibility, testability
- Faktor peralihan produk : portability, reusability, interoperability

Model McCall dan kategori yang diilustrasikan oleh model McCall dari Faktor kualitas perangkat lunak pohon (lihat Gambar 3.1). Tiga bagian berikutnya yang didedikasikan untuk sebuah penjelasan rinci tentang perangkat lunak faktor kualitas termasuk dalam setiap kategori McCall's.

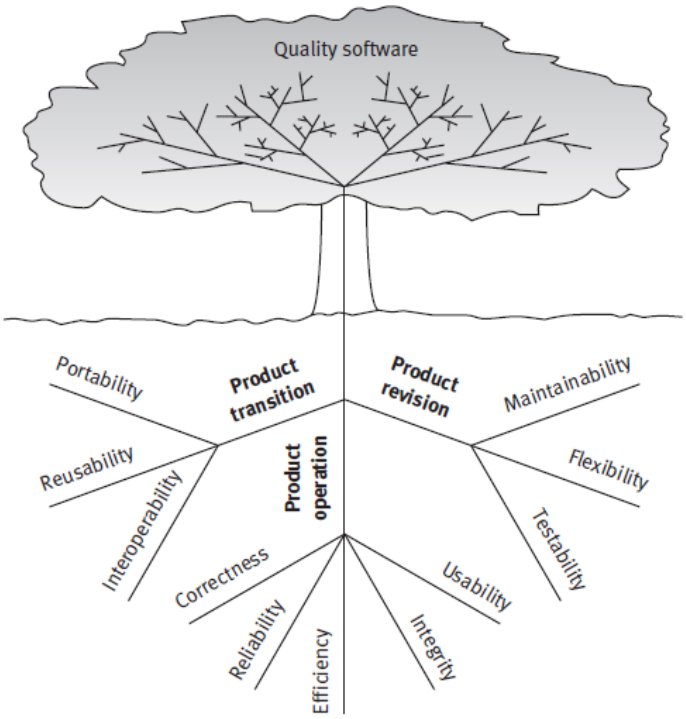


Figure 3.1: McCall's factor model tree

Source: Based on McCall *et al.*, 1977

3.3. Faktor kualitas software untuk operasional produk

- Correctness (kebenaran)

Kebutuhan kebenaran didefinisikan sebagai kebutuhan keluaran dari sebuah sistem software yang tertulis dalam sebuah daftar, seperti tampilan query dari sistem informasi penjualan. Ketetapan output biasanya multidimensi, beberapa diantaranya termasuk :

- Tujuan keluaran
- Akurasi dari keluaran yang dapat dapat merugikan jika perhitungan datanya tidak tepat.
- Kelengkapan dari informasi keluaran yang merugikan jika informasinya tidak lengkap.
- Up to-date informasi (dipertimbangkan sebagai waktu kejadian oleh sistem informasi)
- Ketersediaan informasi (reaksi terhadap waktu, didefinisikan sebagai waktu yang diperlukan untuk mendapatkan informasi yang dibutuhkan sebagai reaksi atas permintaan perusahaan)
- Standarisasi coding dan dokumentasi sistem software

Persyaratan Correctness dari suatu sistem informasi keanggotaan klub adalah sebagai berikut:

- The output mission: Daftar terdefinisi yang terdiri dari 11 jenis laporan, empat jenis huruf standar untuk anggota dan delapan jenis pertanyaan, yang akan ditampilkan pada monitor berdasarkan permintaan.
- Ketepatan yang diharuskan dari output: Probabilitas untuk non-akurat output, mengandung satu atau lebih kesalahan, tidak akan melebihi 1%.
- Kesempurnaan informasi keluaran: Probabilitas hilang data tentang anggota, kehadirannya di acara-acara klub, dan pembayaran-nya tidak melebihi 1%.
- Informasi terbaru: Tidak lebih dari dua hari kerja untuk informasi tentang partisipasi dalam acara dan tidak lebih dari satu hari kerja untuk informasi tentang masuknya pembayaran anggota dan pribadi data.
- Tersedianya informasi: Reaksi waktu untuk kueri akan kurang dari dua detik rata rata, waktu reaksi untuk laporan akan kurang dari empat jam.
- Standar diperlukan dan pedoman: Perangkat lunak dan dokumentasinya diwajibkan untuk mematuhi pedoman klien.

- Reliability (tahan uji)

Kebutuhan akan reliability terkait dengan kegagalan menyediakan layanan. Kebutuhan ini menentukan rata-rata kegagalan yang diperbolehkan dalam sistem dan mengacu ke keseluruhan sistem atau satu atau lebih dari fungsi yang berbeda.

Persyaratan Reliability berurusan dengan kegagalan untuk menyediakan layanan. Mereka menentukan tingkat kegagalan maksimum yang diperbolehkan sistem perangkat lunak, dan dapat merujuk pada seluruh sistem atau untuk satu atau lebih fungsi terpisah.

Contoh

(1) Frekuensi kegagalan unit heart-monitoring yang akan beroperasi dalam bangsal perawatan intensif rumah sakit ini harus kurang dari satu dalam 20 tahun. Fungsi deteksi serangan jantung diwajibkan untuk memiliki tingkat kegagalan kurang dari satu per satu juta kasus.

(2) Salah satu persyaratan dari sistem perangkat lunak baru untuk dipasang di utama cabang Bank Kemerdekaan, yang mengoperasikan 120 cabang, adalah bahwa hal itu tidak akan gagal, rata-rata, lebih dari 10 menit per bulan selama bank kantor jam. Selain itu, probabilitas bahwa off-waktu (waktu yang dibutuhkan untuk perbaikan dan pemulihan semua layanan bank) akan lebih dari 30 menit ini harus kurang dari 0,5%.

- Efficiency

Kebutuhan efficiency terkait dengan kebutuhan akan sumber daya hardware untuk melaksanakan seluruh fungsi dalam sebuah sistem software dan memenuhi semua kebutuhan yang lain.

Efisiensi persyaratan berurusan dengan sumber daya perangkat keras yang diperlukan untuk melakukan semua fungsi sistem perangkat lunak dalam kesesuaian dengan semua persyaratan yang lain. Sumber daya perangkat keras utama yang harus dipertimbangkan adalah komputer kemampuan pengolahan (diukur dalam MIPS - juta instruksi per detik, MHz atau megahertz - juta siklus per detik, dll), data kemampuan penyimpanan dalam hal memori dan kapasitas disk (diukur dalam MBs - megabyte, GBS - gigabyte, TB - Terabytes, dll) dan kemampuan komunikasi data dari jalur komunikasi (biasanya diukur dalam kbps - kilobit per kedua, Mbps - megabit per detik, dan Gbps - gigabit per detik). The Persyaratan mungkin termasuk nilai maksimum di mana hardware sumber daya akan diterapkan dalam sistem perangkat lunak yang dikembangkan atau firmware.

Jenis lain berkaitan dengan kebutuhan efisiensi waktu antara recharging unit portabel sistem, seperti, sistem informasi unit terletak di komputer portabel, atau unit meteorologi ditempatkan di luar ruangan.

- Integrity

Kebutuhan integrity terkait dengan sistem keamanan software yang diperlukan untuk mencegah akses bagi orang yang tidak berhak untuk membedakan antara orang yang boleh melihat informasi sama yang tidak.

- Usability (dapat dipakai/dipergunakan)

Kebutuhan usability terkait dengan cakupan sumber daya karyawan yang diperlukan untuk melatih anggota baru dan mengoperasikan sistem software.

Persyaratan Usability berurusan dengan ruang lingkup sumber daya staf yang dibutuhkan untuk melatih karyawan baru dan untuk mengoperasikan sistem perangkat lunak. Untuk lebih lanjut tentang kegunaan lihat Juristo et al. (2001), Donahue (2001) dan Ferre et al. (2001).

Contoh

Kegunaan perangkat lunak persyaratan dokumen untuk help desk yang baru sistem diprakarsai oleh sebuah perusahaan jasa alat rumah daftar berikut ini

Spesifikasi:

(a) Seorang anggota staf harus mampu menangani paling tidak 60 layanan panggilan sehari.

(b) Pelatihan karyawan baru tidak akan mengambil lebih dari dua hari (16 pelatihan jam), segera pada akhir dimana peserta pelatihan akan mampu menangani 45 layanan panggilan sehari.

3.4. Faktor kualitas software untuk perbaikan produk

Menurut model McCall faktor kualitas perangkat lunak, tiga faktor kualitas terdiri dari kategori revisi produk. Faktor-faktor tersebut berhubungan dengan orang-orang persyaratan yang mempengaruhi lengkap kegiatan pemeliharaan perangkat lunak: pemeliharaan korektif (koreksi kesalahan perangkat lunak dan kegagalan), adaptif pemeliharaan (mengadaptasi perangkat lunak saat ini untuk keadaan tambahan dan pelanggan tanpa mengubah perangkat lunak) dan perfektif pemeliharaan (peningkatan dan perbaikan perangkat lunak yang ada dengan mengenai isu-isu lokal terbatas). Ini adalah sebagai berikut.

Maintainability

Persyaratan Maintainability menentukan upaya yang akan dibutuhkan oleh pengguna dan personil perawatan untuk mengidentifikasi alasan kegagalan perangkat lunak, untuk memperbaiki kegagalan, dan untuk memverifikasi keberhasilan koreksi. Ini persyaratan faktor tersebut mengacu pada struktur modular perangkat lunak, internal program dokumentasi, dan manual programmer, di antara barang-barang lainnya.

Contoh

Khas maintainability persyaratan:

(a) ukuran sebuah modul perangkat lunak tidak akan melebihi 30 laporan.

(b) pemrograman akan mematuhi perusahaan pengkodean standar dan pedoman.

Flexibility

Kemampuan dan upaya yang diperlukan untuk mendukung kegiatan pemeliharaan adaptif tercakup oleh persyaratan fleksibilitas. Ini termasuk sumber daya (yaitu dalam manusia-hari) yang diperlukan untuk mengadaptasi paket perangkat lunak untuk berbagai pelanggan perdagangan yang sama, dari luasan berbagai kegiatan, dari rentang yang berbeda produk dan sebagainya. Persyaratan ini faktor tersebut juga mendukung perfektif pemeliharaan kegiatan, seperti perubahan dan penambahan perangkat lunak dalam rangka meningkatkan layanan dan untuk disesuaikan dengan perubahan dalam perusahaan teknis atau komersial lingkungan.

Contoh

TSS (guru perangkat lunak pendukung) berkaitan dengan dokumentasi prestasi murid, perhitungan nilai akhir, pencetakan dokumen grade panjang, dan pencetakan otomatis surat peringatan kepada orang tua murid gagal. Spesifikasi perangkat lunak termasuk persyaratan fleksibilitas sebagai berikut:

(a) Perangkat lunak ini harus sesuai untuk guru-guru semua mata pelajaran dan sekolah semua tingkatan (SD, SMP dan tinggi).

(b) Non-profesional harus mampu menciptakan jenis baru laporan menurut dengan kebutuhan guru dan / atau pendidikan kota departemen tuntutan.

Testability

Testability persyaratan berurusan dengan pengujian sistem informasi sebagai juga dengan operasi. Testability persyaratan untuk kemudahan pengujian adalah terkait dengan fitur khusus dalam program yang membantu tester, misalnya dengan memberikan hasil antara standar dan file log. Testability persyaratan terkait dengan operasi software otomatis yang dilakukan mencakup diagnostik oleh sistem perangkat lunak sebelum memulai sistem, untuk mengetahui apakah semua komponen dari sistem perangkat lunak yang berfungsi dengan baik dan untuk mendapatkan laporan tentang kesalahan terdeteksi. Tipe lain dari persyaratan transaksi ini dengan cek diagnostik otomatis diterapkan oleh para teknisi pemeliharaan untuk mendeteksi penyebab kegagalan perangkat lunak.

Contoh

Sebuah unit kontrol industri komputer diprogram untuk menghitung berbagai ukuran status produksi, laporan tingkat kinerja mesin, dan mengoperasikan sinyal peringatan dalam situasi yang telah ditetapkan. Satu testability persyaratan dituntut adalah untuk mengembangkan satu set data uji standar dengan diharapkan sistem yang dikenal reaksi yang benar di setiap tahap. Ini uji standar data akan dijalankan setiap pagi, sebelum produksi dimulai, untuk memeriksa apakah unit komputer bereaksi dengan benar.

3.5. Faktor kualitas software untuk peralihan produk

Menurut McCall, tiga faktor kualitas termasuk dalam transisi produk kategori, kategori yang berkaitan dengan adaptasi perangkat lunak untuk lain lingkungan dan interaksi dengan sistem perangkat lunak lain.

Portabilitas

Persyaratan Portabilitas cenderung adaptasi dari sistem perangkat lunak untuk lainnya lingkungan yang terdiri dari hardware yang berbeda, sistem operasi yang berbeda, dan sebagainya. Persyaratan ini memungkinkan untuk terus menggunakan yang sama software dasar dalam situasi beragam atau menggunakannya secara bersamaan di berbagai hardware dan sistem operasi situasi.

Contoh

Sebuah paket perangkat lunak yang dirancang dan diprogram untuk beroperasi di Windows 2000 lingkungan diperlukan untuk memungkinkan transfer murah untuk Linux dan Windows NT lingkungan.

Reusabilitas

Reusability persyaratan berurusan dengan penggunaan perangkat lunak modul awalnya dirancang untuk satu proyek dalam proyek software baru saat ini sedang dikembangkan.

Mereka juga dapat memungkinkan proyek-proyek masa depan untuk menggunakan modul yang diberikan atau grup modul dari software yang dikembangkan saat ini. Penggunaan kembali perangkat lunak diharapkan dapat menghemat sumber daya pembangunan, memperpendek masa pengembangan, dan menyediakan modul lebih berkualitas. Manfaat ini kualitas yang lebih tinggi berdasarkan asumsi bahwa sebagian besar kesalahan perangkat lunak telah terdeteksi oleh kegiatan jaminan kualitas dilakukan pada perangkat lunak yang asli, oleh pengguna perangkat lunak asli, dan selama menggunakan kembali yang sebelumnya. Isu-isu reuse perangkat lunak menjadi subjek standar industri perangkat lunak (lihat IEEE, 1999).

Contoh

Sebuah unit pengembangan perangkat lunak telah diwajibkan untuk mengembangkan sebuah sistem perangkat lunak untuk operasi dan kontrol dari sebuah kolam renang hotel yang melayani hotel tamu dan anggota klub renang. Walaupun manajemen tidak mendefinisikan persyaratan usabilitas, pemimpin tim unit, setelah menganalisis informasi persyaratan pengolahan spa hotel, memutuskan untuk menambahkan persyaratan reusabilitas bahwa beberapa perangkat lunak modul untuk kolam renang

harus dirancang dan diprogram dengan cara yang akan memungkinkan penggunaan kembali dalam spa's perangkat lunak masa depan sistem, yang rencananya akan dikembangkan tahun depan.

Modul ini akan memungkinkan:

■ memeriksa validitas Masuk kartu keanggotaan dan merekam kunjungan.

■ Restaurant penagihan.

■ Pengolahan surat perpanjangan keanggotaan.

Interoperabilitas

Interoperabilitas persyaratan fokus pada pembuatan interface dengan perangkat lunak lain sistem atau dengan firmware peralatan lain (misalnya, firmware dari mesin produksi dan peralatan pengujian antarmuka dengan produksi pengendalian perangkat lunak). Persyaratan Interoperabilitas dapat menentukan nama (s) dari perangkat lunak atau firmware untuk interface mana yang akan diperlukan. Mereka bisa juga menentukan struktur output diterima sebagai standar dalam industri tertentu atau aplikasi daerah.

Contoh

Firmware peralatan laboratorium medis diperlukan untuk prosesnya hasil (output) menurut struktur data standar yang kemudian dapat berfungsi sebagai masukan untuk sejumlah sistem informasi laboratorium standar.

3.6. Alternatif model tentang faktor kualitas software

Dua faktor model, muncul pada akhir 1980-an, dianggap alternatif model McCall faktor klasik (McCall et al, 1977.), pantas diskusi:

- Para Evans dan model Marciniak faktor (Evans dan Marciniak, 1987).
- Para Deutsch dan model Willis faktor (Deutsch dan Willis, 1988).

3.6.1. Formal perbandingan model alternatif

Perbandingan formal model faktor mengungkapkan:

- Kedua model alternatif mengecualikan hanya satu dari 11 McCall's faktor, yaitu faktor testability.
- Para Evans dan model Marciniak faktor terdiri dari 12 faktor yang diklasifikasikan menjadi tiga kategori.
- Para Deutsch dan model Willis faktor terdiri dari 15 faktor yang diklasifikasikan menjadi empat kategori.

Secara keseluruhan, lima faktor baru yang disarankan oleh faktor alternatif dua model:

- Verifiability (oleh kedua model)
- Ekspandibilitas (oleh kedua model)
- Keamanan (oleh Deutsch dan Willis)
- Manageability (oleh Deutsch dan Willis)
- Survivability (oleh Deutsch dan Willis).

Faktor-faktor termasuk dalam model berbagai faktor dibandingkan pada Tabel 3.1.

Faktor tambahan didefinisikan sebagai berikut.

Verifiability (disarankan oleh Evans dan Marciniak)

Persyaratan verifiability mendefinisikan fitur desain dan pemrograman yang memungkinkan verifikasi efisien dari desain dan pemrograman. Kebanyakan verifiability Persyaratan lihat modularitas, untuk kesederhanaan, dan kepatuhan terhadap dokumentasi dan panduan pemrograman.

Upgrade (disarankan oleh Evans dan Marciniak, dan Deutsch dan Willis)

Persyaratan Ekspandibilitas lihat upaya masa depan yang akan diperlukan untuk melayani populasi yang lebih besar, meningkatkan pelayanan, atau menambah aplikasi baru dalam rangka untuk meningkatkan kegunaan. Sebagian besar persyaratan ini dicakup oleh McCall's fleksibilitas faktor.

Keselamatan (disarankan oleh Deutsch dan Willis)

Persyaratan keselamatan dimaksudkan untuk menghilangkan kondisi berbahaya untuk operator peralatan sebagai akibat dari kesalahan dalam proses software kontrol. Ini kesalahan bisa mengakibatkan reaksi yang tidak tepat untuk situasi berbahaya atau ke kegagalan untuk memberikan sinyal alarm bila kondisi berbahaya untuk dideteksi oleh perangkat lunak muncul.

Table 3.1: Comparison of McCall's factor model and alternative models

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak	Deutsch and Willis
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

Contoh

Dalam sebuah pabrik kimia, sistem komputerisasi mengendalikan aliran asam menurut perubahan tekanan dan temperatur yang terjadi selama produksi. The persyaratan keselamatan lihat reaksi komputerisasi sistem dalam kasus situasi yang berbahaya dan juga menentukan apa jenis alarm yang diperlukan dalam setiap kasus.

Manageability (disarankan oleh Deutsch dan Willis)

Persyaratan Manageability mengacu pada perangkat administrasi yang mendukung software modifikasi selama pengembangan perangkat lunak dan pemeliharaan periode, seperti manajemen konfigurasi, perangkat lunak prosedur perubahan, dan sejenisnya.

Contoh

"Chemilog" adalah sistem perangkat lunak yang secara otomatis log arus bahan kimia ke dalam berbagai wadah untuk memungkinkan analisis kemudian efisiensi unit produksi. Perkembangan dan masalah versi baru dan pelepasan

"Chemilog" dikendalikan oleh Dewan Pengembangan Software, yang
anggota bertindak sesuai dengan prosedur modifikasi perangkat lunak perusahaan.

Survivability (disarankan oleh Deutsch dan Willis)
Persyaratan survivabilitas merujuk pada kesinambungan layanan. Ini mendefinisikan waktu minimum diperbolehkan antara kegagalan sistem, dan maksimal waktu yang diizinkan untuk pemulihan layanan, dua faktor yang berkaitan dengan pelayanan kontinuitas. Meskipun persyaratan tersebut dapat merujuk secara terpisah terhadap total dan kegagalan sebagian layanan, mereka terutama diarahkan untuk kegagalan dari esensial fungsi atau jasa. Kesamaan yang signifikan yang ada antara survivabilitas faktor dan faktor Reliability yang dijelaskan dalam model McCall's.

Contoh
Taya mengoperasikan lotere nasional, diadakan sekali seminggu. Sekitar 400 000-700000 taruhan ditempatkan mingguan. Sistem baru perangkat lunak pelanggan (yang Taya National Lottery) telah memerintahkan akan sangat terkomputerisasi dan berdasarkan sistem komunikasi yang menghubungkan semua mesin taruhan ke pusat komputer. Untuk persyaratan lainnya Reliability yang tinggi, Taya telah menambahkan berikut persyaratan survivabilitas: Probabilitas bahwa kerusakan tidak terpulihkan ke file taruhan akan terjadi dalam hal kegagalan sistem akan terbatas pada kurang dari satu dalam satu juta.

3.6.2 Perbandingan model faktor - analisis isi
Setelah membandingkan isi dari model faktor, kita menemukan bahwa dua dari tambahan lima faktor, Ekspandibilitas dan survivability, benar-benar menyerupai faktor yang sudah termasuk dalam model faktor McCall's, meskipun di bawah berbeda nama, Fleksibilitas dan Reliabilitas. Selain itu, faktor Testability McCall bisa dianggap sebagai salah satu unsur dalam faktor rawatan sendiri. Ini berarti bahwa perbedaan antara tiga model faktor yang jauh lebih kecil dari yang dirasakan. Artinya, faktor model alternatif hanya menambahkan tiga "baru" faktor untuk model McCall's:

- Kedua model menambahkan verifiability faktor.
- Para Deutsch dan Willis model menambahkan faktor Keselamatan dan Manageability.

3.6.3 Struktur model faktor alternatif

Namun demikian, meskipun kesamaan mereka, kategori dipekerjakan oleh alternatif faktor model dan klasifikasi faktor-faktor tertentu ke dalam kategori berbeda dari yang ditawarkan oleh model McCall's. Tabel 3.2 membandingkan struktur dari tiga model sesuai dengan faktor-faktor dan klasifikasinya dalam kategori.

3.7. Siapa yang tertarik dalam pendefinisian kebutuhan kualitas

Secara alami, ada pemikiran bahwa hanya seorang pelanggan yang teliti lah yang tertarik untuk mendefinisikan kebutuhannya untuk memastikan kualitas dari produk software yang dipesannya. Dokumen kebutuhan dipersiapkan sebagai dasar agar tidak mendapatkan kualitas yang buruk. Para analisis dari berbagai macam faktor kualitas mengindikasikan bagaimana pengembang software dapat menambah persyaratan yang menunjukkan keinginannya. Berikut beberapa contoh :

1) Kebutuhan reusability

Dalam kasus di mana klien mengantisipasi perkembangan dalam waktu dekat sistem software tambahan yang kuat kesamaan untuk perangkat lunak ini, klien akan dirinya memulai reusabilitas persyaratan. Dalam kasus lain, klien tertarik untuk menggunakan kembali bagian perangkat lunak sistem yang dikembangkan sebelumnya dalam suatu sistem yang baru. Namun, itu lebih mungkin bahwa pengembang, yang melayani berbagai macam klien, akan mengenali potensi manfaat menggunakan kembali, dan akan memasuki reusabilitas ke dalam daftar persyaratan yang harus dipenuhi oleh tim proyek.

2) Kebutuhan verifiablity

Persyaratan ini dimaksudkan untuk meningkatkan desain review dan uji perangkat lunak yang dilakukan selama pengembangan perangkat lunak. Tujuan mereka adalah untuk menghemat sumber daya pembangunan dan mereka, Oleh karena itu, yang menarik bagi pengembang. Klien Namun, biasanya tidak tertarik dalam menempatkan kebutuhan yang berhubungan dengan kegiatan internal tim pengembang.

Beberapa faktor kualitas tidak termasuk dalam persyaratan klien khas dokumen dapat, dalam banyak kasus, bunga pengembang. Daftar berikut kualitas faktor biasanya bunga pengembang sedangkan mereka dapat meningkatkan bunga yang sangat sedikit pada bagian dari klien:

■ Portabilitas

■ Reusability

■ verifiability.

Table 3.2: Comparison of the structure of McCall's factor model *vis-à-vis* the three alternative models

McCall's model categories	Software quality factors	Deutsch and Willis model categories				Evans and Marciniak model categories		
		Functional	Performance	Change	Management	Design	Performance	Adaptation
Product operation	Correctness		x			x		
	Reliability	x					x	
	Efficiency		x				x	
	Integrity	x					x	
	Usability	x					x	
Product revision	Maintainability			x		x		
	Flexibility			x				x
	Testability							
Product transition	Portability			x				x
	Reusability			x				x
	Interoperability		x					x
Factors of the alternative models	Verifiability				x	x		
	Expandability			x				x
	Safety		x					
	Manageability				x			
	Survivability	x						

Jadi, kita dapat berharap bahwa proyek akan dilaksanakan sesuai dengan dua Persyaratan dokumen:

■ Dokumen Persyaratan klien

■ Dokumen tambahan ini pengembang persyaratan.

3.8. Pemenuhan software terhadap faktor kualitas

Selama proses pengembangan perangkat lunak, sejauh mana proses sesuai dengan persyaratan kualitas faktor berbagai diperiksa oleh tinjauan desain, inspeksi perangkat lunak, pengujian perangkat lunak, dan sebagainya sebagainya. Komprehensif diskusi tentang tinjauan desain, pengujian perangkat lunak, perangkat lunak metrik kualitas dan alat lainnya untuk memverifikasi dan memvalidasi kualitas perangkat lunak yang disediakan dalam saldo buku ini. Selain itu, kepatuhan produk perangkat lunak dengan persyaratan milik berbagai faktor kualitas diukur oleh metrik kualitas perangkat lunak, langkah-langkah yang mengukur tingkat kepatuhan. Dalam rangka untuk memungkinkan pengukuran yang valid kepatuhan, sub-faktor telah ditetapkan untuk mereka

Faktor model	Faktor Kualitas Software	Sub Faktor
Model Mc Call's Kategori operasional produk	Correctness (kebenaran)	Akurasi
		Kelengkapan
		Up-to-dates (terbaru)
		Availability (tanggap waktu)
		Koding dan petunjuk dokumentasi

		Pemenuhan dan konsistensi
	Reliability (tahan uji/diandalkan)	Sistem reliable
		Aplikasi reliable
		Pemulihan kegagalan komputasi
		Pemulihan kegagalan hardware
	Efisiensi	Efisiensi dalam pemrosesan
		Efisiensi dalam penyimpanan data
		Efisiensi dalam komunikasi
		Efisiensi dalam penggunaan power
	Integrity (keamanan)	Akses kontrol
		Akses audit
	Usability (digunakan)	Operasional
		Pelatihan

Faktor model	Faktor Kualitas Software	Sub Faktor
Model Mc Call's Kategori perbaikan produk	Maintainability (Pemeliharaan)	Kesederhanaan
		Modular
		Dideskripsikan sendiri
		Koding dan petunjuk dokumentasi
		Pemenuhan (konsistensi)
		Kemudahan mengakses dokumentasi
	Flexibility (kelenturan)	Modular
		General
		Sederhana
		Dideskripsikan sendiri
	Testability (percobaan)	Testing oleh user
		Testing pemeliharaan kegagalan
		Traceability

Faktor model	Faktor Kualitas Software	Sub Faktor
Model Mc Call's	Portability	Kemandirian sistem software
Kategori peralihan produk		Modular
		Pendeskripsian sendiri
	Reusability	Modular
		Kemudahan akses dokumen
		Kemandirian sistem software
		Kemandirian aplikasi
		Pendeskripsian sendiri

Seperti Anda mungkin melihat, beberapa sub-faktor yang berhubungan dengan lebih dari salah satu faktor. Ini mencerminkan fakta bahwa beberapa atribut berkontribusi untuk sukses kepatuhan di lebih dari satu aspek penggunaan perangkat lunak.

Misalnya, kesederhanaan (sub-faktor) memberikan kontribusi untuk rawatan, usabilitas fleksibilitas, dan upgrade faktor.

3.9. Ringkasan

Kebutuhan untuk dokumen persyaratan yang komprehensif dan isinya. Banyak kasus kepuasan pelanggan rendah adalah situasi dimana proyek-proyek perangkat lunak telah memuaskan memenuhi persyaratan dasar correctness, sementara penderitaan dari kinerja miskin di daerah penting lainnya seperti pemeliharaan, kehandalan, perangkat lunak kembali, atau pelatihan. Salah satu penyebab utama penyimpangan ini adalah kurangnya ditetapkan persyaratan yang berkaitan dengan aspek-aspek fungsionalitas perangkat lunak. Oleh karena itu, ada kebutuhan untuk definisi yang komprehensif dari persyaratan yang akan mencakup semua aspek menggunakan perangkat lunak di semua tahapan dari siklus hidup perangkat lunak. Faktor model mendefinisikan spektrum yang luas dari kebutuhan perangkat lunak. Kami berharap bahwa orang-orang yang menetapkan persyaratan perangkat lunak akan mengacu pada setiap faktor dan karenanya, memeriksa kebutuhan untuk menggabungkan persyaratan masing-masing di mereka persyaratan dokumen.

(2) Struktur (kategori dan faktor-faktor) dari model faktor klasik McCall's. Model faktor McCall's mengklasifikasikan kebutuhan semua perangkat lunak menjadi kualitas perangkat lunak 11 faktor. Ke-11 faktor tersebut dikelompokkan menjadi tiga kategori - produk operasi, produk revisi dan transisi produk - sebagai berikut:

- faktor operasi Produk: Ketelitian, Reliability, Efisiensi, Integritas, Usability.
- faktor revisi Produk: Maintainability, Flexibility, Testability.
- faktor transisi Produk: Portabilitas, Reusability, Interoperabilitas.

(3) Faktor-faktor tambahan yang disarankan oleh model faktor alternatif.

Faktor model dua dari akhir 1980-an, alternatif untuk faktor McCall klasik model, adalah:

- Para Evans dan model Marciniak faktor.
- Para Deutsch dan model Willis faktor.

Model-model alternatif menyarankan untuk menambahkan lima faktor untuk model McCall's. Dua dari faktor-faktor ini sangat mirip dengan faktor dua ofMcCall's, hanya tiga faktor yang "baru":

■ Kedua model menambahkan verifiability faktor.

■ Para Deutsch dan Willis model menambahkan faktor Keselamatan dan Manageability.

(4) Mereka yang tertarik dalam menetapkan persyaratan kualitas perangkat lunak. Klien bukan pihak yang hanya tertarik pada menyeluruh mendefinisikan persyaratan yang menjamin kualitas produk perangkat lunak. Pengembang sering tertarik menambahkan persyaratan yang mewakili kepentingan sendiri, seperti usabilitas, verifiability dan portabilitas persyaratan. Ini tidak mungkin, bagaimanapun, akan menarik perhatian para klien. Dengan demikian, kita dapat berharap bahwa proyek akan dilaksanakan sesuai dengan dua Persyaratan dokumen:

■ Dokumen Persyaratan klien

■ Dokumen tambahan ini pengembang persyaratan.

Bab4. Gambaran Komponen Komponen Penjamin Kualitas Perangkat Lunak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan secara utuh bagian-bagian yang menggambarkan komponen pada tiap tahapan yang berbeda.
2. Memberi penjelasan secara sepintas hal-hal yang harus dipenuhi dan rekomendasinya.

Bab ini merupakan akhir dari pendahuluan tentang SQA yang memberikan gambaran yang luas tentang komponen SQA yang disediakan bagi para pembuat rencana dalam sebuah sistem SQA intra organisasi. Sebagai sebuah sistem lokal, sistem SQA antar organisasi memuat warna dasar yang dipengaruhi oleh karakteristik dari organisasi. Proyek pengembangannya, kegiatan pemeliharaan software dan staf profesional. Ringkasan singkat dari komponen SQA diberikan melalui diskusi tentang pertimbangan panduan pembangunan sistem organisasi SQA. Pandangan sekilas ini akan memperkenankan kita untuk mendapatkan pemahaman awal tentang potensi kontribusi dari masing-masing komponen, tentang keseluruhan jangkauan komponen dan tentang sistem yang didefinisikan sebagai sebuah entitas.

4.1. Sistem SQA dan Arsitektur SQA

Sebuah sistem SQA selalu mengkombinasikan banyak komponen SQA, semuanya digunakan untuk menantang banyaknya kode sumber dari kesalahan software dan mendapatkan tingkat penerimaan dalam software yang berkualitas.

Komponen sistem SQA dapat dikelompokkan dalam enam bagian yaitu :

a. Komponen Pra-Proyek

- Untuk meyakinkan bahwa komitmen proyek telah cukup memadai untuk didefinisikan dengan mempertimbangkan sumber daya yang dibutuhkan, jadwal yang ada dan biaya yang dianggarkan.
- Perencanaan pengembangan dan perencanaan mutu pekerjaan telah ditentukan dengan benar

b. Komponen Penilaian Kegiatan Siklus Hidup Proyek

Siklus hidup proyek disusun menjadi dua yaitu tahap siklus hidup pengembangan dan tahap operasional dan pemeliharaan. Komponen pada tahap siklus hidup pengembangan memeriksa rancangan dan kesalahan program. Komponen ini selanjutnya dibagi menjadi empat sub-kelas sebagai berikut :

- Review
- Pendapat ahli
- Ujicoba software
- Pemeliharaan software
- Jaminan kualitas atas subkontraktor dan bagian penyedia barang lain

Komponen SQA digunakan selama tahap operasional dan pemeliharaan termasuk khususnya komponen pemeliharaan seperti komponen siklus hidup pengembangan yang diterapkan sebagian besar untuk tugas peningkatan pemeliharaan fungsional.

c. Komponen Infrastruktur untuk Pencegahan Kesalahan dan Perbaikan

Tujuan utama dari komponen ini, yang diterapkan seluruhnya oleh organisasi, adalah untuk menghilangkan atau setidaknya mengurangi jumlah kesalahan berdasarkan akumulai pengalaman SQA yang dimiliki organisasi.

d. Komponen Manajemen Kualitas Software

Komponen dari kelas ini dicocokkan terhadap beberapa tujuan, salah satu yang utama adalah pengawasan terhadap kegiatan pengembangan dan pemeliharaan serta memperkenalkan dukungan pengelolaan standar yang utamanya mencegah kesalahan penjadwalan dan penganggaran biaya dan keluarannya.

e. Komponen Standarisasi, Sertifikasi dan Penilaian Sistem SQA

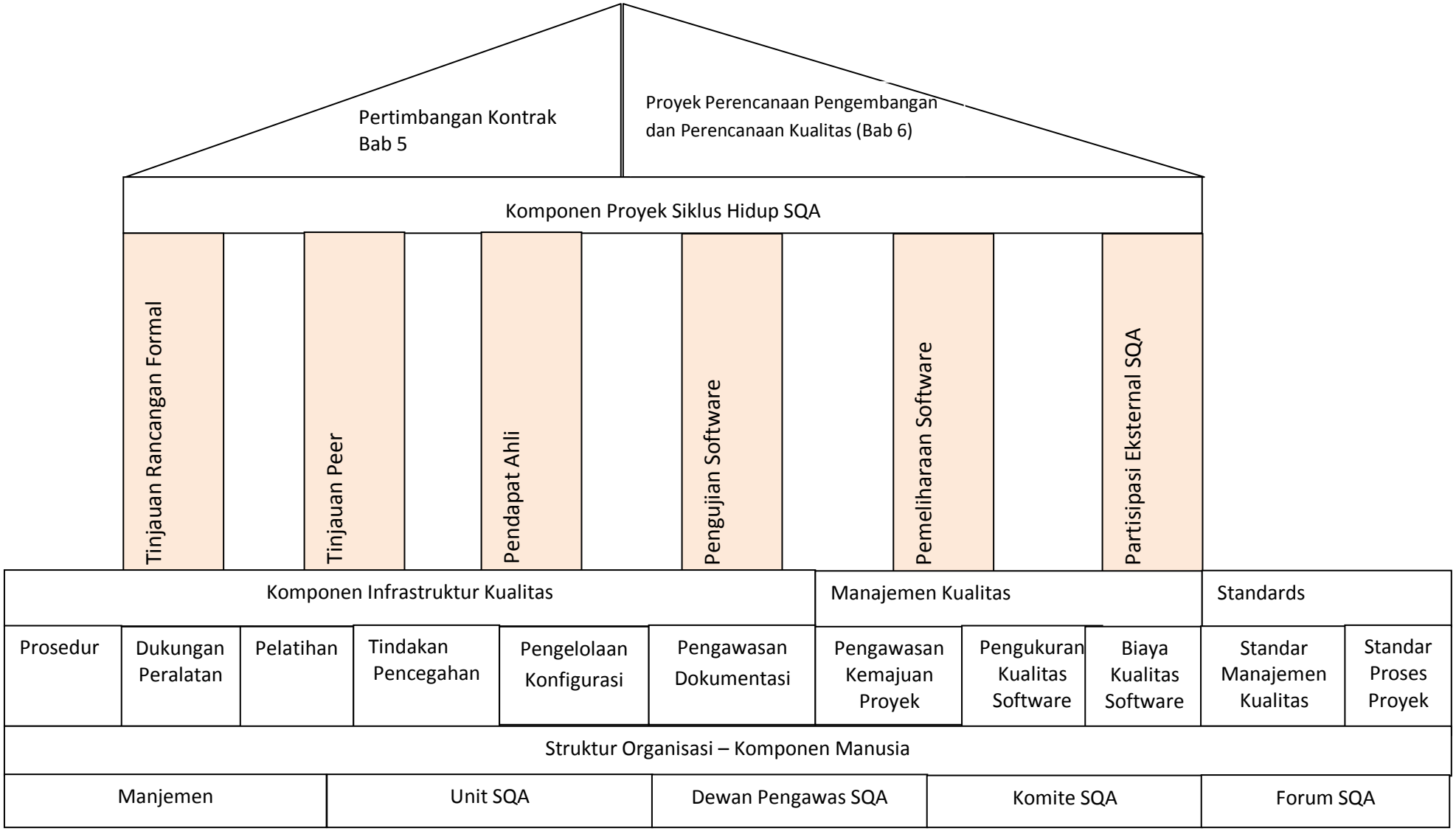
Komponen ini memperkenalkan keahlian level internasional dan standar pengelolaan dalam sebuah organisasi. Tujuan utama dari kelas ini adalah : (a)pemanfaatan standar keahlian tingkat internasional, (b)peningkatan koordinasi antara sistem kualitas organisasi dengan organisasi lain, (c) penilaian pencapaian kualitas sistem berdasarkan skala yang umum. Macam-macam standar dapat diklasifikasikan menjadi dua group utama yaitu : (i) standar pengelolaan kualitas, (ii) standar proses proyek.

f. Komponen Manusia untuk Mengorganisasi SQA

Pada dasarnya organisasi SQA melibatkan manajer, karyawan tester, unit SQA dan praktisi yang tertarik terhadap kualitas software (dewan pengawas SQA, anggota komite SQA, anggota forum SQA). Semua aktor yang berkontribusi terhadap kualitas software, tujuan utamanya adalah memprakarsai dan mendukung penerapan komponen SQA, memeriksa penyimpangan terhadap prosedur SQA dan metodologinya serta saran perbaikan.

Frame 4.1 Klasifikasi komponen sistem SQA

- Komponen kualitas pra proyek
- Komponen kualitas siklus hidup proyek
- Komponen infrastrukur pencegah kesalah dan perbaikan
- Komponen pengelolaan kualitas software
- Komponen standarisasi, sertifikasi dan penilaian SQA
- Komponen manusia – mengorganisasi SQA


Pertimbangan Kontrak
Bab 5
Proyek Perencanaan Pengembangan
dan Perencanaan Kualitas (Bab 6)
Komponen Proyek Siklus Hidup SQA
Tinjauan Rancangan Formal
Tinjauan Peer
Pendapat Ahli
Pengujian Software
Pemeliharaan Software
Partisipasi Eksternal SQA
Komponen Infrastruktur Kualitas
Manajemen Kualitas
Standards
Prosedur
Dukungan Peralatan
Pelatihan
Tindakan Pencegahan
Pengelolaan Konfigurasi
Pengawasan Dokumentasi
Pengawasan Kemajuan Proyek
Pengukuran Kualitas Software
Biaya Kualitas Software
Standar Manajemen Kualitas
Standar Proses Proyek
Struktur Organisasi – Komponen Manusia
Manjemen
Unit SQA
Dewan Pengawas SQA
Komite SQA
Forum SQA

4.2. Komponen Pra-Proyek

Komponen SQA dalam tahap ini digunakan untuk meningkatkan tahap persiapan berdasarkan prakarsa awal dalam proyek.

1.2.1 Review Kontrak

Review kontrak melibatkan detil pemeriksaan yang terdiri dari : (a) draft proposal proyek, (b) draft kontrak. Untuk review kontrak khususnya melibatkan kegiatan :

- Klarifikasi kebutuhan pelanggan
- Review jadwal proyek dan perkiraan kebutuhan akan sumber daya
- Evaluasi keahlian staf untuk mengerjakan usulan proyek
- Evaluasi kemampuan pelanggan untuk memenuhi kewajiban
- Evaluasi resiko tahap pengembangan

Pendekatan yang mirip diterapkan dalam kontrak pemeliharaan, seperti pembenahan kesalahan, pemeliharaan layanan termasuk adaptasi software dan keterbatasan kegiatan pengembangan software demi peningkatan performa yang diistilahkan dengan “peningkatan pemeliharaan fungsional”

1.2.2 Perencanaan Pengembangan dan Mutu

Persoalan utama yang dipelihara dalam perencanaan pengembangan proyek adalah :

- Jadwal
- Orang yang berpengaruh dan sumber daya hardware
- Evaluasi resiko
- Persoalan organisasi : anggota tim, sub kontraktor dan kemitraan
- Metodologi proyek, tool pengembangan
- Rencana penggunaan kembali software

Persoalan utama yang dipelihara dalam perencanaan mutu proyek adalah :

- Kualitas yang dituju, diungkapkan dalam istilah terukur yang sesuai
- Kriteria awal dan akhir dari tiap tahapan proyek
- Daftar uji pemeriksaan, uji coba, verifikasi jadwal lain dan validasi kegiatan

4.3. Komponen Siklus Hidup Proyek Software

Siklus hidup proyek dibagi menjadi dua tingkatan yaitu : tingkat siklus hidup pengembangan dan tingkat operasional dan pemeliharaan. Beberapa komponen SQA memasuki siklus hidup proyek pengembangan software melalui beberapa titik yang berbeda. Penggunaan komponen ini seharusnya direncanakan sebelum memulai proyek. Komponen-komponen tersebut diantaranya :

- Review
- Pendapat para ahli
- Uji coba software
- Pemeliharaan software
- Kepastian kualitas dari subkontraktor dan suplier

4.3.1. Review

Tahap rancangan pada proses pengembangan menghasilkan bermacam-macam dokumen. Produk yang tercetak termasuk rancangan laporan, dokumen uji coba software, perencanaan installasi software dan panduan software. Review dapat dikategorikan menjadi dua macam yaitu : Design Review (DR) dan Peer Review.

Design Review Formal

Bagian penting dari dokumen-dokumen ini membutuhkan persetujuan formal dari profesional dalam bidang kualitas sebagai penetapan dalam kontrak pengembangan dan permintaan prosedur untuk diterapkan oleh pengembang software. Hal ini harusnya ditekankan bahwa pengembang hanya dapat melanjutkan ke tahap berikutnya dari proses pengembangan pada dokumen yang telah disetujui secara formal.

Laporan DR itu sendiri meliputi daftar dari perbaikan yang dibutuhkan. Ketika sebuah komite review rancangan duduk untuk memutuskan keberlangsungan sebuah proyek, maka salah satu pilihan yang dapat dipertimbangkan adalah :

- Persetujuan segera dari dokumen DR dan melanjutkan ke fase pengembangan berikutnya.
- Persetujuan untuk memproses ke fase pengembangan berikutnya sesudah semua proses telah dilengkapi dan diperiksa oleh komite yang mewakili.
- Tambahan DR diperlukan dan dijadwalkan untuk mengambil tempat sesudah semua proses telah dilengkapi dan diperiksa oleh komite yang mewakili.

Peer Review

Peer review (pemeriksaan) diarahkan pada pemeriksaan dokumen secara singkat, bab atau bagian dari sebuah laporan, code modul program software yang dicetak dan sejenisnya. Pemeriksaan dapat menggunakan beberapa formulir dan beberapa method.

4.3.2. Pendapat Ahli

Pendapat ahli sering kali digunakan untuk mendukung penilaian kualitas dengan cara penambahan kemampuan eksternal terhadap organisasi yang memiliki proses pengembangan. Peralihan ke ahli luar bisa jadi bagian yang berguna untuk beberapa situasi berikut :

- Kemampuan profesional personel organisasi tidak cukup dalam area tertentu
- Sebuah organisasi kecil dalam banyak kasus kesulitan untuk menemukan kandidat yang pantas untuk diikutsertakan dalam tim review rancangan. Di beberapa situasi ahli dari luar bergabung dalam komite DR.
- Dalam sebuah organisasi kecil atau dalam situasi yang ekstrim maka pendapat ahli bisa menggantikan pemeriksaan

4.3.3. Uji Coba Software

Uji coba software merupakan komponen SQA formal yang merupakan target pemeriksaan ketika menjalankan software dalam posisi benar-benar berjalan. Uji coba ini berdasarkan daftar yang telah dipersiapkan sebelumnya yang menyajikan beberapa gambaran skenario. Uji coba software memeriksa modul software, integrasi software, keseluruhan paket software (sistem). Perbaikan uji coba terhadap perbaikan dari kesalahan yang ditemukan dilanjutkan sampai pelanggan merasa puas terhadap hasil yang didapat. Tujuan langsung dari uji coba software atau pendeteksian kesalahan software untuk memenuhi persyaratan adalah persetujuan formal dari sebuah modul atau aturan integrasi sehingga fase program berikutnya dapat dimulai atau dilengkapi sehingga bisa dikirim dan diinstall.

Beberapa program testing dibuat dari bermacam-macam uji coba, beberapa diantaranya secara manual dan lainnya otomatis. Semua tes telah dirancang, direncanakan dan disetujui berdasarkan prosedur pengembangan. Laporan uji coba termasuk dalam detil pemeriksaan dan menjadi rekomendasi

tentang performa dari sebagian atau keseluruhan uji coba mengikuti bagian dari perbaikan berdasarkan penemuan uji coba.

4.3.4. Komponen Pemeliharaan Software

Layanan pemeliharaan software luas cakupannya dan disediakan dalam periode waktu lama biasanya beberapa tahun. Layanan ini terbagi menjadi beberapa kategori sebagai berikut :

- *Corrective maintenance*, layanan yang digunakan untuk mendukung user, melakukan perbaikan terhadap kesalahan kode maupun kesalahan dokumentasi.
- *Adaptive maintenance*, adaptasi dari software terhadap lingkungan dan pelanggan baru tanpa perubahan mendasar dari produk software. Adaptasi ini biasanya dibutuhkan ketika sistem hardware atau komponen mengalami modifikasi
- *Functionality improvement maintenance*, peningkatan fungsi dan performa yang terkait dengan kondisi software yang ada, dikeluarkan dengan memenuhi beberapa batasan masalah.

Layanan pemeliharaan software seharusnya memenuhi semua jenis persyaratan kualitas, sebagian berupa fungsionalitas dan kebutuhan akan penjadwalan (biasanya ditentukan bersama dengan pelanggan) sebagaimana pula batasan biaya (ditentukan oleh penyedia jasa). Ketentuan terhadap pemeliharaan layanan yang akan berjalan melibatkan aplikasi dari sebagian besar macam komponen SQA. Komponen SQA yang digunakan dalam menjamin kualitas terhadap pemeliharaan sistem adalah sebagai berikut :

- Komponen pra-proyek
    - Review kontrak pemeliharaan
    - Perencanaan pemeliharaan
- Komponen siklus hidup pengembangan software

  Komponen ini diterapkan untuk meningkatkan fungsionalitas dan adaptasi pemeliharaan, kegiatan yang memiliki karakteristik yang mirip dengan proses pengembangan software.
- Komponen infrastruktur SQA
    - Prosedur pemeliharaan dan petunjuk instruksi
    - Peralatan pendukung kualitas
    - Pelatihan karyawan pemelihara, pelatihan ulang dan sertifikasi.
    - Pemeliharaan pencegahan dan tindakan perbaikan

- Pengelolaan konfigurasi
- Pengawasan terhadap dokumentasi pemeliharaan dan pencatatan kualitas

- Komponen pengelolaan pengawasan SQA
  - Pengawasan pemeliharaan layanan
  - Pengukuran pemeliharaan kualitas
  - Biaya pemeliharaan kualitas

4.3.5. Jaminan kualitas dari kerja partisipan luar

Subkontraktor dan pelanggan sering bergabung dengan pengembang langsung dikontrak (Yang "pemasok") dalam melaksanakan proyek-proyek pengembangan perangkat lunak. Jika lebih besar dan lebih kompleks proyek, semakin besar kemungkinan bahwa pihak luar akan diperlukan, dan semakin besar proporsi pekerjaan yang akan dilakukan kepada mereka (subkontraktor, pemasok perangkat lunak COTS dan pelanggan). Motivasi untuk mengubah kepada pihak luar terletak pada apapun.

4.4. Komponen Infrastruktur untuk mencegah eror dan perbaikan

Komponen SQA dari kelas ini meliputi :

- Prosedur dan petunjuk kerja
- Template dan daftar pemeriksaan
- Pelatihan, pelatihan berkelanjutan dan sertifikasi
- Tindakan pencegahan dan perbaikan
- Pengelolaan konfigurasi
- Pengawasan dokumentasi

4.4.1. Prosedur dan Petunjuk Kerja

Prosedur jaminan kualitas biasanya menyediakan definisi yang lengkap terhadap pelaksanaan tipe khusus dari kegiatan pengembangan dengan sebuah cara yang memastikan pencapaian yang efektif terhadap hasil kualitas. Prosedur direncanakan agar dapat diterapkan secara umum dan melayani keseluruhan organisasi. Petunjuk kerja, dilain pihak menyediakan arahan yang lengkap untuk menggunakan metode yang diterapkan dalam instansi khusus dan dikerjakan oleh tim khusus juga.

Prosedur dan petunjuk kerja didasarkan pada kumpulan pengalaman dan pengetahuan sebuah organisasi seperti kontribusi mereka terhadap pelaksanaan teknologi yang benar dan efektif. Karena

bercermin pada pengalaman sebelumnya dengan memperhatikan serta memperbaharui dan menyesuaikan beberapa prosedur dan instruksi, terorganisasi dan kondisi-kondisi lainnya.

4.4.2. Peralatan Pendukung Kualitas

Salah satu cara untuk mengkombinasikan kualitas yang lebih tinggi dengan efisiensi yang lebih adalah menggunakan dukungan peralatan kualitas seperti template dan daftar pemeriksaan. Peralatan ini berdasarkan akumulasi dari pengetahuan dan pengalaman mereka terhadap pengembangan dan pemeliharaan organisasi secara profesional, berkontribusi terhadap tujuan SQA sebagai berikut :

- Menghemat waktu yang diperlukan untuk mendefinisikan struktur dari bermacam-macam dokumen atau daftar persiapan dari subyek yang akan diperiksa
- Berkontribusi terhadap kelengkapan dokumen dan pemeriksaan
- Peningkatan komunikasi diantara tim pengembang dan anggota komite pemeriksaan dengan cara menstandarkan dokumen dan agenda.

4.4.3. Pelatihan, Petunjuk dan Sertifikasi

Pernyataan biasa yang terlatih dan intruksi professional staff yang baik adalah kunci untuk efisiensi , kualitas pekerjaan, tidak membuat observasi, tidak lebih benar. Dengan *framework* dari SQA, menjaga sumber daya manusia organisasi berpengetahuan dan merubah penyimpanan level dicapai terutama dengan :

- Pelatihan pegawai baru dan *retrining* beberapa pekerja yang mendapatkan kenaikan pangkat.
- Melanjutkan perubahan staff dengan memperhatikan pengembangan professional dan *in-house,* sesuai dengan keahlian yang dipunyai.
- Sertifikasi pegawai setelah pengetahuan dan kemampuan mereka ditunjukkan

4.4.4. Tindakan Pencegahan dan Perbaikan

Pelajaran Sistematik dari koleksi data mengenai *instances* dari kesalahan dan kesuksesan kontribusi untuk kualitas jaminan proses dengan beberapa jalan, yaitu :

- Implementasi dari perubahan yang mencegah kesalahan yang sama di kemudian hari
- Koreksi dari beberapa kesalahan sama yang ditemukan di project lain dan terdiri dari performa aktivitas oleh team lain

- Mengimplementasikan metodologi yang terbukti berhasil untuk menambah probabilitas dari keberhasilan yang terulang.

4.4.5. Pengelolaan Konfigurasi

Pengembangan Software secara teratur dan pemeliharaan operasi melibatkan aktivitas yang intensif dimana modifikasi software untuk membuat versi baru dan mereleasenya. Perbedaan anggota team membuatt aktivitas ini menstimulasi, walaupun mereka mungkin meletakkan di tempat yang berbeda. Hasilnya, bahaya serius timbul, meskipun dari kehilangan identifikasi, atau kehilangan dari dokumentasi. Konsekuensi kesalahan mungkin terjadi.

Konfigurasi manajemen terjadi dengan beberapa resiko dari pengenalan produk untuk mengontrol perubahan proses. Prosedur ini dihubungkan untuk mengizinkan adanya perubahan, menurut dari versi dan spesifikasi dari software yang di install dan penanganan dari beberapa perubahan versi. Banyak konfigurasi manajemen system mengimplementasikan peralatan komputerisasi untuk meng-*compile* pekerjaan mereka. Software konfigurasi prosedur biasanya mengautorisasi admin atau konfigurasi manajemen komite untuk mengatur semua tugas konfigurasi manajemen operasi.

4.4.6. Pengawasan Dokumentasi

Fungsi pengawasan dokumentasi utamanya mengacu pada kebutuhan dokumen pelanggan, dokumen kontrak, laporan perancangan, perencanaan proyek, standar pengembangan dan lain sebagainya. Kegiatan pengawasan dokumentasi memerlukan :

- Definisi dari tipe pengawasan dokumen yang diperlukan
- Spesifikasi format, metode identifikasi dokumen dan lain-lain.
- Definisi dari pemeriksaan dan proses persetujuan untuk masing-masing dokumen pemeriksaan
- Definisi dari metode penyimpanan pencapaian

4.5. Pengelolaan Komponen SQA

Komponen pengelolaan SQA mendukung pengawasan pengelolaan terhadap proyek pengembangan software dan layanan pemeliharaan. Komponen pengawasan melibatkan :

- Pengawasan kemajuan proyek
- Pengukuran kualitas software
- Biaya kualitas software

•

4.5.1. Pengawasan Kemajuan Proyek

Kegiatan pengawasan proyek fokus pada :

- Penggunaan sumber daya
- Penjadwalan
- Kegiatan pengelolaan resiko
- Pembiayaan

4.5.2. Pengukuran Kualitas Software

Diantara pengukuran kualitas software yang ada atau masih dalam proses pengembangan, kita dapat mendata pengukuran untuk beberapa hal berikut :

- Kualitas pengembangan software dan kegiatan pemeliharaan
- Produktivitas tim pengembang
- Produktivitas help desk dan tim pemeliharaan
- Tingkat kepadatan kesalahan software
- Penyimpangan jadwal

4.5.3. Biaya Kualitas Software

Kualitas harga dipengaruhi oleh pengembangan software dan aplikasi, perpanjangan model kualitas harga, harga dari control, digabungkan dengan harga dari kesalahan. Manajemen secara khusus dihasilkan dari jumlah total dari kualitas harga. Sehingga dipercaya bahwa kenaikan ke tingkat tertentu, memperluas sumber daya yang dialokasikan untuk mengontrol aktifitas hasil yang lebih besar menghemat biaya kegagalan sekaligus mengurangi kualitas total.

Analisis kualitas harga dapat membantu identifikasi anggota yang tidak memiliki jaminan kualitas efektif, upaya menghasilkan rata-rata kualitas harga yang lebih tinggi. Hasilnya dapat digunakan membantu memperbaiki keanggotaan.

4.6. Standar SQA, Sistem Sertifikasi dan Komponen Penilaian

Peralatan eksternal menawarkan kesempatan lain untuk mencapai tujuan tentang kepastian kualitas software. Tujuan utama dari komponen kelas ini adalah :

- Penggunaan pengetahuan internasional secara profesional
- Peningkatan koordinasi dengan sistem kualitas dari organisasi lain.
- Evaluasi tujuan secara profesional dan pengukuran terhadap pencapaian sistem kualitas organisasi

Standar yang tersedia diklasifikasikan menjadi dua sub kelas yaitu : standar pengelolaan kualitas dan standar proses proyek. Salah satu ataupun kedua sub kelas diperlukan oleh pelanggan dan ditetapkan dalam persetujuan kontrak yang disertakan.

4.6.1. Standar Pengelolaan Kualitas

Standar paling umum yang dikenal adalah :

- SEI CMM standar penilaian
- Standar ISO 9001 dan ISO 9000-3

4.6.2. Standar Proses Proyek

Standar yang digunakan :

- IEEE 1012 Standard
- ISO/IEC 12207 Standard

4.7. Organisasi SQA – Komponen Manusia

Tujuan utama dari organisasi SQA adalah sebagai berikut :

- Untuk mengembangkan dan mendukung penerapan komponen SQA
- Untuk mendeteksi penyimpangan dari prosedur SQA dan metodologi
- Untuk memberikan saran perbaikan terhadap komponen SQA

4.7.1. Pengelolaan Peran di SQA

Tanggung jawab dari top manajemen, manajemen departemen, proyek manajemen meliputi :

- Mendefinisikan kebijakan kualitas
- Tindakan lanjutan yang efektif terhadap penerapan kebijakan kualitas

- Mengalokasikan sumber daya yang cukup untuk menerapkan kebijkan kualitas
- Penetapan staf yang memadai
- Tindak lanjut dari pemenuhan prosedur jaminan kualitas
- Solusi terhadap penjadwalan, pembiayaan dan kesulitan hubungan dengan pelanggan.

4.7.2. Unit SQA

Unit ini dan software penguji adalah satu-satunya bagian dari dasar SQA organisasi yang mengabdikan diri penuh waktu untuk permasalahan SQA. Semua sekmen lainnya dari SQA dasar organisasi berkontribusi hanya beberapa waktu mereka untuk masalah kualitas. Dengan begitu, Unit tugas SQA ini disediakan sebagai *moving force* utama, inisiator, dan coordinator dari SQA system dan aplikasi.

Tugas ini dapat dipecah menjadi beberapa peran utama :

- Persiapan program kualitas tahunan
- Konsultasi dengan staf internal dan ahli dari luar terhadap persoalan kualitas software
- Melaksanakan audit internal jaminan kualitas
- Kepemimpinan dari beberapa komite penjamin kualitas

4.7.3. Dewan Pengawas, Komite dan Forum SQA

Kontribusinya meliputi :

- Tim penyelesaian persoalan atau unit permasalahan kualitas kecil
- Memeriksa penyimpangan terhadap prosedur kualitas dan petunjuk kerja
- Memprakarsai peningkatan dalam komponen SQA
- Pelaporan ke unit SQA tentang persoalan kualitas dalam tim mereka atau unitnya.

Persoalan utama terkait dengan komite :

- Jalan keluar terhadap permasalahan kualitas software
- Analisis persoalan dan kesalahan pencatatan seperti pencatatan hal yang lain, diikuti dengan prakarsa pembenahan dan pencegahan yang sesuai.
- Prakarsa dan pengembangan prosedur dan petunjuk baru, mengupdate materi baru.
- Prakarsa dan pengembangan komponen SQA baru dan peningkatan komponen yang ada

4.8. Pertimbangan Pedoman Pembuatan Sistem Organisasi SQA

Keputusan berdasarkan sistem pengelolaan kualitas software sebuah perusahaan dapat dipecah menjadi dua bagian yaitu :

- Berdasarkan organisasi SQA
- Komponen SQA diterapkan dalam organisasi dan bisa diperluas penggunaannya.

Keputusan ini dipengaruhi sejumlah pertimbangan mendasar yang mencerminkan karakteristik dari (a) organisasi, (b) proyek pengembangan software dan layanan pemeliharaan untuk ditampilkan dan (c) profesionalitas staf organisasi.

Pertimbangan organisasi :

- Tipe para pelanggan dari pengembangan software

  Kemungkinan para pelanggan termasuk para pembeli dari paket software, pelanggan dari software customisasi dan pelanggan internal (dari organisasi itu sendiri beserta unit maupun sub-unit)
- Tipe para pelanggan dari pemeliharaan software

  Para pelanggan pemeliharaan boleh berbeda secara subtansial dari para pelanggan pengembangan software. Sebagai contoh sebuah unit pemelihara internal mungkin melayani pembelian paket software maupun software customisasi khususnya yang dikembangkan oleh software house dari sebuah organisasi. Juga sebuah software house mungkin memperkerjakan sub kontraktor untuk melakukan pemeliharaan paket software yang dijual ke pelanggan selama waktu garansi dan sesudahnya.
- Range produk

  Kondisi yang mungkin merubah dari lingkup luas produk menjadi lingkup terbatas untuk produk maupun layanan khusus saja.
- Ukuran dari organisasi

  Ukuran biasa yang diterapkan pada sebuah organisasi adalah jumlah pekerja profesional. Secara umum, semakin banyak jumlah pekerja profesional yang dipekerjakan maka semakin banyak spesialisasi yang dimiliki maka semakin banyak pula komponen SQA yang dikembangkan dan diterapkan.

- Derajat dan kealamian kerjasama dengan organisasi lain menyelesaikan proyek yang berhubungan

  Lingkup pilihan kerjasama yang tersedia
- Mengoptimalkan tujuan

  Organisasi diperlukan untuk memilih komponen SQA dalam mendapatkan laporan kontribusi yang digabungkan dengan optimal dalam area berikut : (a) kualitas software, (b)tim produksi, (c) efisiensi proses, (d) penghematan keuangan.

Pertimbangan layanan proyek dan pemeliharaan :

- Level kompleksitas dan kesulitan software
- Tingkat pengalaman staf terhadap teknologi proyek
- Perluasan penggunaan ulang software dalam proyek baru

Pertimbangan profesional staf :

- Kualifikasi tingkat profesionalitas
- Tingkat pengetahuan terhadap anggora tim

4.9. Ringkasan

Frame 4.2

Pertimbangan utama yang mempengaruhi penggunaan komponen SQA.

Pertimbangan Organisasi :

- Tipe para pelanggan dari pengembangan software
- Tipe para pelanggan dari pemeliharaan software
- Range produk
- Ukuran dari organisasi
- Derajat dan kealamian kerjasama dengan organisasi lain menyelesaikan proyek yang berhubungan
- Mengoptimalkan tujuan

Pertimbangan Layanan Proyek dan Pemeliharaan

- Level kompleksitas dan kesulitan software
- Tingkat pengalaman staf terhadap teknologi proyek
- Perluasan penggunaan ulang software dalam proyek baru

Pertimbangan profesional staf :

- Kualifikasi tingkat profesionalitas
- Tingkat pengetahuan terhadap anggora tim

Bab5. Pemeriksaan Kontrak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Mampu menjelaskan dua tahapan pemeriksaan kontrak.
2. Mendata tujuan dari masing-masing tahapan pemeriksaan kontrak.
3. Mengidentifikasi faktor-faktor yang mempengaruhi perluasan kontrak.
4. Mengindentifikasi kesulitan dalam pelaksanaan pemeriksaan kontrak mayor
5. Menjelaskan beberapa rekomendasi yang dapat dilakukan terhadap sebuah pemeriksaan mayor
6. Mendiskusikan pentingnya melakukan pemeriksaan kontrak terhadap proyek yang sifatnya internal.

Kontrak yang tidak jelas/ jelek merupakan awal kegiatan yang tidak menyenangkan. Dari sudut pandang SQA kontrak yang jelek biasanya mempunyai karakter tidak lengkap dalam hal pendefinisian kebutuhan dengan jadwal dan biaya yang tidak realistis yang diperkirakan akan menghasilkan software dengan kualitas rendah. Sehingga secara alami program SQA dimulai dengan pencegahan dalam hal penjaminan kualitas dengan cara mereview draft proposal dan selanjutnya draft kontrak (keduanya biasanya dijadikan satu kegiatan yaitu review kontrak). Kedua review tersebut bertujuan untuk memperbaiki pembiayaan dan penjadwalan yang merupakan dasar dan bagian dari sebuah kontrak dan menampakkan potensi perangkap pada tahap awal (ada dalam draft proposal dan draft kontrak).

Bab ini ditujukan untuk mempelajari tujuan dari review kontrak dan luasnya cakupan dari subyek yang direview yang berhubungan dengan tujuannya. Proses review kontrak dimulai dengan hubungan pelanggan dan penyedia yang diharapkan membuat kontribusi yang penting terhadap internal proyek.

Setelah mempelajari bab ini, anda diharapkan mampu :

- Menjelaskan dua tingkatan pemeriksaan kontrak
- Menyatakan tujuan dari masing-masing tingkatan pemeriksaan kontrak
- Mengidentifikasi faktor yang berpengaruh terhadap luasnya cakupan pemeriksaan
- Mengidentifikasi kesulitan dalam melakukan pemeriksaan kontrak utama

- Mendiskusikan nilai penting dari hasil pemeriksaan kontrak untuk internal proyek

5.1. Pendahuluan

Review kontrak merupakan bagian dari kualitas software yang mengurangi kemungkinan terjadinya situasi yang tidak menyenangkan. Review kontrak merupakan persyaratan dalam standar ISO 9001 dan ISO 9000-3.

5.2. Proses Pemeriksaan Kontrak dan Tingkatan

Beberapa situasi dapat membuat sebuah perusahaan software menandatangi sebuah kontrak dengan seorang pelanggan. Situasi itu diantaranya :

1) Peserta dalam sebuah tender
2) Mengirimkan sebuah proposal berdasarkan RFP yang diberikan pelanggan
3) Menerima permintaan dari perusahaan pelanggan
4) Menerima permintaan internal dalam organisasi itu sendiri atau pesanan dari unit lain dalam organisasi

Proses review itu sendiri diadakan dalam dua tingkatan yaitu :

- Tingkat 1 – Pemeriksaan draft proposal yang sebelumnya dikirim ke pelanggan potensial
- Tingkat 2 – Pemeriksaan draft kontrak sebelum ditandatangani

5.3. Tujuan Pemeriksaan Kontrak

Seperti yang diharapkan sebelumnya, bahwa dua tahapan review kontrak mempunyai tujuan yang berbeda dengan detil sebagai berikut :

5.3.1. Tujuan Pemeriksaan Draft Proposal

Tujuan pemeriksaan draft proposal adalah meyakinkan bahwa hal berikut ini telah memuaskan :

1) Kebutuhan pelanggan telah diklarifikasi dan didokumentasikan
2) Pendekatan alternatif untuk menyelesaikan proyek telah diperiksa
3) Aspek formal terhadap hubungan antara pelanggan dan perusahaan software telah ditetapkan.
4) Mengidentifikasi resiko pengembangan
5) Perkiraan yang memadai tentang sumber daya dan waktu yang diperlukan

6) Pemeriksaan terhadap kemampuan perusahaan dalam hal penilaian terhadap proyek
7) Pemeriksaan terhadap kemampuan pelanggan untuk memenuhi komitmen
8) Pendefinisian tentang patner dan partisipan subkontraktor
9) Pendefinisian dan perlindungan terhadap hak cipta

5.3.2. Tujuan Pemeriksaan Draft Kontrak

Tujuan pemeriksaan draft kontrak adalah meyakinkan bahwa beberapa kegiatan berikut telah memuaskan :

1) Tidak ada sisa permasalahan yang belum dijelaskan
2) Semua pemahaman telah dicapai oleh pelanggan dan perusahaan secara menyeluruh dan terdokumentasi dengan baik di kontrak dan appendix nya.
3) Pemahaman ini berarti untuk memecahkan sesuatu yang tidak jelas dan berbeda antara pelanggan dan perusahaan telah ditampilkan lebih lanjut.

5.4. Pelaksanaan Pemeriksaan Kontrak

Pemeriksaan kontrak bisa berubah dalam jangkauan yang luas tergantung pada tujuan proyek. Kerumitan ini kemungkinan berasal dari sisi teknikal maupun organisasi. Tingkat perbedaan usaha profesional di suguhkan dalam berbagai macam pemeriksaan kontrak. Upaya khusus profesional diperlukan untuk proposal utama.

5.4.1. Faktor yang berpengaruh terhadap perluasan pemeriksaan kontrak

Faktor proyek terpenting yang menentukan besarnya usaha review kontrak yang diperlukan adalah :

- Besarnya proyek, sumber daya biasanya diukur dalam orang bulan
- Kerumitan teknikal proyek
- Tingkat kedalaman pengetahuan staf dan sisi pengalaman terhadap lingkup proyek. Kedalaman ini diukur terhadap tingkat keseringan terkait kemungkinan penggunaan ulang software, dalam beberapa kasus bagian dari penggunaan kembali software adalah memungkinkan, luasnya lingkup review bisa dikurangi.

Oleh karena itu kita bisa mengasumsikan bahwa proyek yang sederhana dapat dilakukan seorang reviewer tetapai proyek yang besar bisa dilakukan oleh satu tim yang memeriksa luasnya subyek yang dikerjakan, termasuk permintaan penggunaan waktu dalam jam untuk mengerjakan hal tersebut.

5.4.2. Siapa yang melakukan pemeriksaan kontrak

Tugas untuk melakukan review kontrak bisa dilakukan siapa saja, daftar di bawah ini menunjukkan kemungkinan yang melakukan review tergantung dari kerumitan proyek :

- Ketua tim proposal
- Anggota dari tim proposal
- Profesional dari luar atau anggota staf perusahaan yang bukan anggota dari tim proposal
- Tim yang merupakan kumpulan para ahli dari luar perusahaan. Biasanya tim pemeriksa kontrak terdiri dari para ahli luar yang dilibatkan, khususnya untuk proyek besar. Para ahli dari luar kemungkinan dilibatkan juga dalam pemeriksa kontrak dalam organisasi pengembang software yang kecil karena tidak menemukan orang yang tepat dalam organisasi.

5.4.3. Penerapan Pemeriksaan Kontrak untuk Proposal Utama

Proposal mayor adalah adalah proposal untuk proyek yang mempunyai karakter sedikitnya ada beberapa dari hal berikut : proyek yang sangat besar, kerumitan teknikal nya tinggi, hal baru bagi perusahaan, dari organisasi besar yang rumit yang terdiri dari beberapa perusahaan seperti terdapat : patner, sub kontraktor, pelanggan). Penerapan proses pemeriksaan kontrak untuk proyek mayor biasanya menyebabkan kesulitan sekalipun bagi organisasi besar.

Beberapa hal yang membuat kesulitan dalam melakukan review kontrak khususnya untuk proposal mayor adalah :

- Tekanan waktu
- Pemeriksaan kontrak yang benar membutuhkan kerja profesional yang benar
- Potensi anggota tim yang melakukan pemeriksaan kontrak sangat sibuk

Rekomendasi untuk penerapan pemeriksaan kontrak mayor :

- Pemeriksaan kontrak seharusnya dijadwalkan
- Sebuah tim seharusnya membagikan isi dari daftar pemeriksaan kontrak
- Menentukan ketua tim pemeriksa kontrak

5.5. Subyek Pemeriksaan Kontrak

Pemeriksaan kontrak memeriksa beberapa hal berdasarkan tujuan pemeriksaan kontrak. Daftar periksa merupakan hal peralatan yang berguna untuk menolong tim dalam memeriksa, mengorganisasi kerja tim dan tujuan pencapaian yang relevan dengan subyek. Hal ini jelas bagi beberapa subyek yang terdapat dalam daftar proyek khusus tertentu. Pada waktu yang sama bahkan daftar periksa yang meliputi banyak hal mungkin tidak memasukkan beberapa subyek penting yang relevan untuk diberikan pada sebuah proposal proyek. Hal ini merupakan tugas dari tim pemeriksa kontrak, khususnya ketua tim untuk menentukan daftar subyek yang berhubungan dengan proposal proyek khusus.

Daftar dari subyek review kontrak diklasifikasikan berdasarkan tujuan review kontrak terdapat dalam :

- Appendik 5A : review draft proposal – subyek daftar periksa
- Appendik 5B : review draft kontrak – subyek daftar periksa

5.6. Pemeriksaan Kontrak untuk Proyek Internal

Sebagian besar, jika bukan mayor, proyek software merupakan proyek internal yang dibuat oleh sebuah unit dalam organisasi untuk unit lain dalam organisasi yang sama. Dalam beberapa kasus unit pengembangan software merupakan penyedia bagi unit lain yang berperan sebagai konsumen. Ciri khas proyek internal dan konsumen dalam perusahaan sendiri terdapat pada tabel 5.1

Tabel 5.1. Tipe Proyek Internal dan Pelanggan Internal

Tipe Proyek Internal	Pelanggan Internal	Contoh Proyek
Administrasi atau software operasional untuk diterapkan internal	Administrasi dan unit operasional	○ Sistem Penjualan dan Persediaan ○ Sistem Manajemen Keuangan ○ Sistem Manajemen SDM
Paket software asli yang diperuntukkan untuk dijual ke umum sebagai paket jual	Software departemen marketing	○ Software permainan ○ Software pendidikan ○ Pengolah kata ○ Paket sistem manajemen penjualan dan persediaan
Perusahaan yang ditanamkan ke produk perusahaan	Departemen pengembangan produk elektronik dan mesin	○ Produk pengatur (control) dan instrumentasi elektronik ○ Mesin dan peralatan untuk rumah tangga dan hiburan ○ Peralatan mainan canggih

Seringnya proyek-proyek ini merupakan persetujuan umum, dimana persetujuan merupakan peran utama dalam hubungan antar dua unit. Jika mengikuti hal itu, unit pengembangan hanya akan menyajikan pemeriksaan kontrak yang singkat atau menengah.

Sayangnya dalam hal ini, penjadwalan, sumber daya dan pengembangannya mengandung resiko, sebagai hasilnya beberapa persolaan akan muncul sebagai berikut :

- Tidak cukupnya pendefinisian kebutuhan proyek
- Perkiraan yang tidak tepat terhadap kebutuhan sumber daya
- Penjadwalan yang tidak tepat
- Kesadaran yang tidak cukup terhadap resiko pengembangan

Kesempatan untuk menghindari persoalan diatas dapat ditingkatkan dengan cara menerapkan prosedur untuk mendefinisikan :

- Proposal yang cukup memadai untuk proyek internal
- Penerapan proses review kontrak yang benar untuk proyek internal
- Pemahaman yang cukup antara pelanggan internal dan penyedia internal

Tabel 5.2. Kerugian akibat kurangnya hubungan terhadap proyek internal

Subyek	Kerugian pelanggan internal	Kerugian pengembang internal
Tidak cukupnya pendefinisian kebutuhan proyek	Penyimpangan terhadap kebutuhan aplikasi sehingga menghasilan kepuasan yang rendah	Perubahan kebutuhan lebih tinggi daripada rata-rata; Membuang sumber daya untuk perubahan yang tidak terelakkan
Lemahnya perkiraan terhadap kebutuhan sumber daya	Harapan yang tidak realistik terhadap pengerjaan proyek	Penyimpangan yang besar dari biaya pengembangan; Perselisihan antar unit akibat penambahan biaya
Lemahnya penjadwalan	Terlambat dalam pendistribusian produk baru	Kegiatan pengembangan dibawah tekanan dan cenderung mengabaikan kualitas; Keterlambatan penyelesaian proyek menyebabkan penundaan untuk pekerjaan berikutnya
Kurang sadarnya terhadap resiko pengembangan	Pelanggan tidak siap terhadap resiko dan konsekuensi nya	Keterlambatan pada permulaan menimbulkan kesulitan berikutnya

5.7. Ringkasan

1. Jelaskan dua tingkatan pemeriksaan kontrak !
2. Sebutkan tujuan dari pemeriksaan kontrak !
3. Jelaskan faktor yang berpengaruh terhadap perluasan dari pemeriksaan kontrak !
4. Sebutkan kesulitan dalam melakukan pemeriksaan kontrak untuk proyek mayor !
5. Jelaskan rekomendasi dalam menerapkan pemeriksaan kontrak mayor !
6. Diskusikan apa arti penting menggunakan pemeriksaan kontrak bagi proyek internal !

Appendik A. Draft Review Proposal dan Subyek Ceklist

Tujuan Review	Subyek Review
1. Keperluan/kebutuhan pelanggan telah diuraikan dan didokumentasikan	1) Kebutuhan fungsional 2) Lingkungan operasional pelanggan (hardware, sistem komunikasi data, sistem operasi dll) 3) Interface yang diperlukan dengan paket software lain dan instrumen lain. 4) Kebutuhan terhadap performansi termasuk beban kerja yang didefinsikan sebagai jumlah dari pengguna dan karakteristik penggunaannya. 5) Reliability (kehandalan) sistem 6) Usability (kemudahan) sistem dinyatakan dalam waktu pelatihan yang dibutuhkan untuk seorang operator mencapai kerja yang diinginkan. Jumlah pelatihan dan upaya petunjuk kerja yang dilakukan termasuk jumlah peserta training, trainer, lokasi dan lama waktu. 7) Jumlah instalasi software yang dilakukan oleh supplier termasuk lokasi. 8) Periode garansi, tambahan suplier, metode penyedia dukungan 9) Proposal untuk layanan syarat pemeliharaan, perluasan dari waktu garansi dan kondisinya. 10) Kelengkapan dari semua kebutuhan tender, termasuk informasi tentang tim proyek, sertifikasi dan dokumen yang lain.
2. Pendekatan alternatif dalam rangka menerima proyek yang telah diperiksa dan diuji.	1) Menyatukan antara software yang diguna ulang dan yang dibeli 2) Patner 3) Pelanggan telah menjalankan kegiatan secara internal pengembangan dari beberapa tugas proyek. 4) Sub kontraktor 5) Perbandingan yang cukup memadai sebagai alternatif.

Tujuan Review	Subyek Review
3. Aspek formal dari hubungan antara pelanggan dengan perusahaan software yang telah ditetapkan.	1) Koordinasi dan komite pengawas bersama termasuk prosedurnya. 2) Daftar dokumentasi yang harus diserahkan. 3) Tanggung jawab pelanggan terhadap persediaan awal tentang fasilitas data dan menjawab pertanyaan dari tim 4) Persyaratan dari fase persetujuan oleh pelanggan dan prosedur persetujuan. 5) Partisipasi pelanggan (perluasan dan prosedur) dalam pemeriksaan kemajuan, pemeriksaan rancangan dan testing. 6) Prosedur untuk menangani permintaan perubahan dari pelanggan selama pengembangan dan pemeliharaan, termasuk metode untuk penghitungan biaya perubahan. 7) Kriteria dari selesainya proyek, metode dari persetujuan dan penerimaan. 8) Prosedur untuk menangani keluhan pelanggan dan masalah yang terjadi setelah pekerjaan diterima, termasuk ketidaksesuaian terhadap spesifikasi yang ditentukan yang terjadi setelah periode garansi. 9) Kondisi untuk pemberian bonus bagi proyek yang berakhir lebih cepat dan hukuman jika ada keterlambatan. 10) Kondisi yang harus dipenuhi penetapan keuangan jika sebagian atau seluruh proyek dibatalkan atau sementara dihentikan atas keinginan dari pelanggan. 11) Kondisi ketentuan layanan selama periode garansi 12) Layanan pemeliharaan software dan kondisi, termasuk kewajiban pelanggan untuk mengupdate versi software setiap kali ada permintaan supplier.
4. Identifikasi dari resiko pengembangan	1) Resiko mengguna ulang modul software atau bagian yang membutuhkan kemahiran dari seorang profesional. 2) Resiko terhadap tidak terpenuhinya kebutuhan komponen hardware dan software berdasarkan jadwal.

Tujuan Review	Subyek Review
5. Perkiraan yang memadai terhadap sumber daya dan jadwal	1) Orang-hari yang diperlukan untuk masing-masing fase proyek dan biaya mereka. Apakah perkiraan tersebut termasuk cadangan sumber daya yang digunakan menangani perbaikan termasuk pemeriksaan rancangan, testing dan lain sebagainya. 2) Apakah perkiraan orang-hari termasuk pekerjaan yang diperlukan untuk menyiapkan dokumentasi, khususnya dokumentasi yang akan diserahkan ke pelanggan. 3) Jumlah tenaga manusia yang diperlukan untuk memenuhi kewajiban garansi dan biayanya. 4) Apakah jadwal proyek termasuk waktu yang dibutuhkan untuk melakukan pemeriksaan, testing dll dan membuat perbaikan yang diperlukan.
6. Pemeriksaan terhadap kemampuan perusahaan dalam melaksanakan proyek	1) Kumpulan dari beberapa profesional dalam bidang pengetahuan 2) Tersedianya staf spesialis (dalam waktu/jadwal dan jumlah yang tepat) 3) Tersedianya komputer dan fasilitas pengembangan yang lain dalam jadwal dan jumlah yang tepat 4) Kemampuan menguasai kebutuhan pelanggan dengan menggunakan tool pengembangan khusus dan standar pengembangan software. 5) Garansi dan kewajiban layanan pemeliharaan software dalam jangka waktu lama.

Tujuan Review	Subyek Review
7. Pemeriksaan terhadap kemampuan pelanggan dalam memenuhi kewajibannya.	1) Kemampuan keuangan, termasuk pembayaran kontrak dan tambahan investasi internal 2) Dukungan terhadap semua data dan tanggung jawab terhadap permintaan staf yang mereka munculkan. 3) Penerimaan dan pelatihan bagi karyawan baru maupun lama 4) Kemampuan untuk melengkapi semua tugas tepat waktu dan sesuai dengan kualitas yang diharapkan
8. Definisi dari patner dan kondisi dari partisipan sub kontraktor	1) Pemberian tanggung jawab terhadap kelengkapan tugas terhadap patner, subkontraktor atau pelanggan, termasuk jadwal dan metode koordinasi. 2) Pemberian pembayaran termasuk bonus dan pinalti diantara patner. 3) Jadwal pembayaran sub kontraktor termasuk bonus dan pinalti 4) Jaminan kualitas terhadap kerja yang dilakukan oleh sub kontraktor, patner dan pelanggan termasuk partisipan dalam kegiatan SQA. (perencanaan, pemeriksaan, testing dll)
9. Definisi dan perlindungan terhadap hak milik software	1) Mengamankan hak milik terhadap software yang dijual dari pihak lain. 2) Mengamankan hak milik terhadap file data yang dibeli dari pihak lain 3) Mengamankan hak milik terhadap penggunaan ulang di masa mendatang terhadap software yang dikembangkan dalam proyek customisasi. 4) Mengamankan hak milik software (termasuk : data file) yang dikembangkan oleh perusahaan dan sub kontraktornya selama periode pengembangan dan ketika masih digunakan secara reguler oleh pelanggan.

Appendik B Pemeriksaan Draft Kontrak dan Subyek Ceklist

Tujuan Review	Subyek Review
• Tidak ada sisa persoalan yang tidak jelas	○ Kewajiban supplier didefinisikan dalam draft kontrak dan apendiksnya ○ Kewajiban pelanggan didefinisikan dalam draft kontrak dan apendiknya.
• Semua pemahaman terhadap bagian proposal yang akan dicapai harus didokumentasi dengan benar	○ Pemahaman tentang kebutuhan fungsional proyek ○ Pemahaman tentang persoalan keuangan, termasuk jadwal pembayaran, bonus dan pinalti. ○ Pemahaman terhadap kewajiban pelanggan. ○ Pemahaman terhadap kewajiban patner dan subkontraktor termasuk persetujuan supplier dengan pihak luar.
• Tidak ada perubahan, penambahan dan penghilangan baru yang telah dimasukkan dalam draft kontrak.	○ Draft kontrak lengkap, tidak ada bagian kontrak yang hilang. ○ Tidak perubahan, penambahan, penghilangan terhadap dokumen yang telah disetujui, atas dasar persoalan keuangan, jadwal proyek atau kewajiban dari pelanggan dan patner.

Bab6. Rencana Pengembangan Produk dan Kualitas Produk

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan tujuan dari perencanaan pengembangan dan perencanaan kualitas
2. Menyebutkan unsur perencanaan pengembangan
3. Menyebutkan unsur perencanaan kualitas
4. Menyebutkan macam-macam resiko software yang utama
5. Menjelaskan proses dari manajemen resiko software
6. Mendiskusikan kepentingan dari pengembangan dan perencanaan kualitas terhadap proyek kecil
7. Mendiskusikan kepentingan dari pengembangan dan perencanaan kualitas terhadap proyek internal

6.1. Tujuan Rencana Pengembangan dan Jaminan Kualitas Software

Perencanaan sebagai sebuah proses, memiliki beberapa tujuan yang masing-masing diantaranya berarti mempersiapkan pondasi yang kokoh untuk beberapa hal berikut :

1) Jadwal kegiatan pengembangan yang memimpin kesuksesan dan penyelesaian tepat waktu terhadap sebuah proyek dan perkiraan terhadap jumlah sumber daya manusia dan biayanya.
2) Perekrutan anggota tim dan alokasi sumber daya untuk pengembangan termasuk kegiatan penjadwalan, penggunaaan sdm membutuhkan perkiraan.
3) Memecahkan resiko pengembangan
4) Penerapan kegiatan yang dibutuhkan SQA
5) Menyediakan manajemen dengan data yang diperlukan untuk pengawasan proyek.

6.2. Unsur Rencana Pengembangan

Berdasarakan materi proposal yang diajukan rencana pengembangan proyek disiapkan untuk memenuhi tujuan proposal. Unsur dibawah, berbeda aplikasi berbeda komponen proyek, sebuah rencana pengembangan proyek terdiri dari :

1) Produk proyek

Rencana pengembangan meliputi produk sebagai berikut :

- Dokumen rencana yang menetapkan tanggal penyelesaian, menunjukkan item yang akan diserahkan ke pelanggan (deliverable).
- Produk software (menetapkan tanggal penyelesaian dan tempat installasi)
- Pelatihan (menetapkan tanggal, peserta dan tempat)

2) Interface proyek
   - Interface dengan paket software yang ada
   - Interface dengan tim pengembangan software atau hardware yang berbeda yang bekerja untuk sistem atau proyek yang sama (kerjasama, koordinasi)
   - Interface dengan hardware yang ada

3) Metodologi proyek dan tool pengembangan untuk diterapkan pada tiap tahap proyek
4) Standar dan pengembangan software

   Sebuah daftar (standar pengembangan software dan prosedur) yang seharusnya disiapkan untuk diterapkan dalam sebuah proyek.
5) Pemetaan terhadap proses pengembangan

   Pemetaan proses pengembangan melibatkan penyediaan definisi yang lengkap dari masing-masing tahapan proyek. Definisi ini termasuk input dan output dan rencana kegiatan yang khusus. Definisi kegiatan meliputi :
   - Perkiraan lama kegiatan, perkiraan ini tergantung pada proyek sebelumnya.
   - Rangkaian logika pada masing-masing kegiatan yang diperlihatkan termasuk deskripsi dari ketergantungan kegiatan sebelumnya yang harus dilengkapi.
   - Tipe dari sumber daya profesional yang diperlukan dan perkiraan bagaimana sumber daya ini diperlukan untuk masing-masing kegiatan.
6) Milestone (kejadian penting) proyek

   Untuk masing-masing kegiatan penting, termasuk waktu dan produk proyek (kode dan dokumentasi) yang didefinisikan.
7) Organisasi staf proyek

- Struktur organisasi : definisi dari tim proyek dan tugas mereka, termasuk anggota tim lain seperti pekerja sub kontraktor sementara.
- Kebutuhan profesional : sertifikasi profesional, pengalaman dalam bahasa pemrograman khusus atau tool pengembangan, pengalaman dalam bidang produk software tertentu, tipe dan lain sebagainya.
- Jumlah anggota tim yang diperlukan untuk masing-masing periode waktu, berdasarkan kegiatan yang telah dijadwalkan. Hal ini diharapkan bahwa tim akan memulai kegiatan mereka pada waktu yang berbeda dan ukuran tim bisa berbeda untuk tiap periode, tergantung rencana kegiatan
- Nama dari ketua tim dan anggota. Kesulitan yang mungkin timbul dalam penetapan anggota tim bisa karena perubahan yang tidak bisa diantisipasi dalam penetapannya sebelumnya. Oleh karena itu nama dari staf diperlukan untuk membantu menjaga jalur dari partisipasi mereka sebagai anggota tim.

8) Fasilitas pengembangan

   Fasilitas pengembangan diperlukan meliputi : hardware, software dan tool pengembangan hardware, ruang kantor dan item yang lain. Untuk masing-masing fasilitas, periode waktu yang diperlukan harus ditunjukkan dalam tabel waktu.

9) Resiko pengembangan

   Pada dasarnya resiko pengembangan meliputi :

   - Celah teknologi
   - Kekurangan staf
   - Ketergantungan antar elemen organisasi.

10) Metode pengawasan

    Untuk mengawasi penerapan proyek, manajer proyek dan manajemen menerapkan rangkaian kegiatan pengawasan saat menyiapkan laporan progress dan rapat koordinasi.

11) Perkiraan biaya proyek

    Perkiraan biaya proyek didasarkan pada perkiraan biaya proposal, diikuti oleh review menyeluruh dari relevansi berdasarkan perkiraan sumber daya manusia, negosiasi kontrak dengan subkontraktor dan pemasok. Sebagai contoh, bagian dari proyek direncanakan harus dilaksanakan oleh sebuah tim

developer internal, dilakukan oleh subkontraktor karena tidak adanya tim. Perubahan ini biasanya ada dalam anggaran tambahan substansial.

6.3. Unsur Perencanaan Kualitas

1) Daftar tujuan kualitas
2) Perencanaan pemeriksaan kegiatan
3) Perencanaan uji coba software
4) Perencanaan uji coba untuk pengembang software eksternal
5) Pengelolaan prosedur dan pengelolaan tool

6.4. Rencana Pengembangan dan Rencana Kualitas untuk Proyek Kecil dan Proyek Internal

6.4.1. Proyek Kecil

- Apakah sebuah proyek hanya membutuhkan 40 hari kerja dikerjakan oleh satu orang profesional dan selesai seluruhnya dalam 12 minggu, suguhkan perhitungan investasi terhadap orang-hari dalam menyiapkan skala penuh pengembangan dan perencanaan kualitas ?
- Apakah proyek dapat dilaksanakan oleh tiga profesional dimana total investasinya 30 orang-hari kerja dan selesai dalam 5 minggu, membutuhkan rencana skala perencanaan penuh ?

Perencanaan pengembangan :

- Produk proyek, mengindikasikan hasil yang diserahkan “deliverables”
- Proyek benchmarks
- Resiko pengembangan
- Perkiraan biaya proyek

Beberapa keuntungan merencanakan proyek kecil ketimbang proyek yang tidak terencana telah diidentifikasi :

- Pemahaman yang meliputi banyak hal tentang pekerjaan tercapai.
- Tanggung jawab yang lebih tinggi untuk kewajiban rapat dapat dipenuhi.
- Akan lebih mudah bagi manajemen dan pelanggan untuk membagi pengawasan sebuah proyek dan mengidentifikasikan penundaan yang tidak terduga pada awalnya.

- ❖ Pemahaman yang lebih baik dengan penghormatan yang lebih baik pula terhadap kebutuhan dan timetable dapat dicapai diantara pengembang dan pelanggan.

Hal yang perlu diperjelas adalah perencanaan dan pengembangan kualitas yang diterapkan pada proyek skala besar tidak harus otomatis juga diterapkan pada proyek kecil. Prosedur khusus diperlukan untuk menentukan bagaimana menjaga proyek bisa dilakukan dengan menjawab pertanyaan dibawah untuk :

6.4.2. Proyek Internal

Proyek internal merupakan proyek yang digunakan untuk sebuah departemen dalam organisasi itu sendiri. Dan nantinya akan digunakan ke seluruh perusahaan.

KEUNTUNGAN BAGI TIM DEVELOPER :

- Mencegah anggaran yang berlebihan.
- Mencegah kegagalan proyek akibat penundaan.
- Mencegah kehilangan pasar seperti jatuhnya reputasi

KEUNTUNGAN BAGI PELANGGAN :

- Deviasi lebih kecil
- Pengawasan lebih baik terhadap proses pengembangan.
- Resiko lebih kecil akibat keterlambatan internal.

KEUNTUNGAN BAGI ORGANISASI :

- Mengurangi resiko kehilangan pasar.
- Mengurangi resiko atas keterlambatan supply, mengurangi penalti terhadap kontrak.
- Mengurangi resiko dari kegagalan reputasi.
- Mengurangi resiko permintaan anggaran dana untuk pemasukan(supply tambahan).

6.5. Ringkasan

1) Jelaskan tujuan dari pengembangan dan perencanaan kualitas :
   - ❖ Menghasilkan perencanaan penjadwalan
   - ❖ Mendapatkan anggota tim dan menentukan fasilitas pengembangan
   - ❖ Memecahkan resiko pengembangan

- ❖ Menerapkan kegiatan yang diperlukan dalam SQA
- ❖ Menyediakan manajemen dengan data yang diperlukan untuk pengawasan proyek

2) Sebutukan elemen-elemen rencana pengembangan :
   - ❖ Produk proyek
   - ❖ Interface proyek
   - ❖ Metodologi proyek dan tool pengembangan
   - ❖ Prosedur dan standar pengembangan software
   - ❖ Pemetaan proses pengembangan
   - ❖ Mile stone proyek
   - ❖ Organisasi staf proyek
   - ❖ Fasilitas pengembangan yang diperlukan
   - ❖ Resiko pengembangan
   - ❖ Metodologi pengawasan
   - ❖ Perkiraan biaya proyek

3) Sebutkan elemen-elemen perencanaan kualitas :
   - ❖ Tujuan kualitas
   - ❖ Perencanaan kegiatan pemeriksaan
   - ❖ Perencanaan uji coba software
   - ❖ Perencanaan uji coba penerimaan untuk pengembang software eksternal
   - ❖ Perencanaan pengelolaan konfigurasi

4) Sebutkan jenis resiko utama pembuatan software
   - ❖ Celah teknologi
   - ❖ Kekurangan staf
   - ❖ Ketergantungan antar organisasi : supplier, sub kontraktor dll

5) Jelaskan proses pengelolaan resiko software :

Kegiatan yang terlibat dalam pengelolaan resiko adalah : perencanaan, penerapan dan pengawasan terhadap penerapan. Bagian kegiatan perencanaan adalah identifikasi SRI, evaluasi terhadap SRI dan perencanaan RMA untuk memecahkan masalah tentang SRI.

6) Diskusikan keuntungan dari menyiapkan pengembangan dan perencanaan kualitas untuk proyek kecil!

   Untuk pengembangan proyek kecil (kurang dari 15 orang-hari), persiapan pengembangan dan perancangan kualitas adalah pilihan. Keuntungan utama dari persiapan perencanaan adalah meningkatkan pemahaman pengembang terhadap tugas dan komitmen yang lebih besar untuk menyelesaikan proyek yang telah direncanakan. Sebagai tambahan dokumen perencanaan berkontribusi untuk memberikan pemahaman yang lebih baik antara pengembang dan pelanggan dan lebih mudah serta efektif untuk pengawasan proyek.

7) Diskusikan keuntungan dari menyiapkan pengembangan dan perencanaan kualitas untuk proyek internal.

   Rekomendasi terhadap proyek internal, mengerjakan pekerjaan dari departemen lain dan untuk mengembangkan paket software dicocokkan dengan pasar ke depan, dirawat sebagai “proyek reguler”. Hal ini menyatakan bahwa skala penuh pengembangan dan perencanaan kualitas disiapkan. Keuntungan yang akan diperoleh meliputi :

   - Departemen pengembangan : akan terhindar dari kerugian yang akan datang akibat tabel waktu dan biaya yang tidak realistik, konsekuensi merusak proyek lain dan reputasi perusahaan.
   - Pelanggan internal : akan mengurangi resiko dari terlambatnya penyelesaian dan biaya yang lebih dan meningkatkan pengawasan proyek serta koordinasi dengan pengambang.
   - Perusahaan : mengurangi resiko terhadap keterlambatan produk software ke pasar, mengurangi resiko yang mengurangi reputasi dalam menghasilkan produk terhadap keterlambatan penyediaan dan mengurangi resiko dari biaya yang membengkak.

6.6. Apendik 6A. Resiko Pengembangan Perangkat Lunak dan Pengelolaan Manajemen Resiko

6A.1. Resiko Pengembangan Perangkat Lunak

Beberapa daftar risiko potensial pengembangan perangkat lunak ("item risiko perangkat lunak"atau SRIS) disebutkan dalam literatur. Ropponen dan Lyytinen (2000) telah mengklasifikasikan item perangkat lunak risiko ke dalam enam kelas berikut :

1. Penjadwalan dan resiko waktu
2. Sistem fungsi resiko
3. Resiko Subkontrak
4. Kebutuhan manajemen resiko
5. Sumber Daya penggunaan dan kinerja resiko
6. Personalia manajemen resiko.

Boehm dan Ross (1989) menyarankan daftar 10 item risiko software utama. Tabel 6A.1 menunjukkan bagaimana daftar ini dapat diintegrasikan dengan enam kelas risiko yang diusulkan oleh Ropponen dan Lyytinen (2000). Metodologi untuk identifikasi barang risiko perangkat lunak telah ditawarkan oleh Boehm (1991), Keil et al., (1998), Ropponen dan Lyytinen (2000), Barki et al, (1993). dan IEEE (2001). Salah satu alat yang paling efektif untuk mengidentifikasi dan mengevaluasi risiko perangkat lunak item daftar khusus, juga disebutkan oleh beberapa penulis. Karolak (1996) dan Jones (1994) telah memperluas ruang lingkup item risiko perangkat lunak untuk memasukkan risiko strategis, seperti risiko pemasaran dan risiko keuangan. Penulis ini berpendapat bahwa meskipun pentingnya risiko strategis, mereka berada di luar cakupan jaminan perangkat lunak yang berkualitas dan dengan demikian di luar cakupan buku ini.

Tabel 6A.1. Daftar 10 Resiko Perangkat Lunak Teratas.

No.	Perangkat Lunak risiko kelas (Ropponen dan Lyytinen)	No.	Perangkat Lunak risiko item (Boehm dan Ross)	Deskripsi
1	Personil manajemen risiko	1	Personil kekurangan	Kekurangan dan pergantian personil yang memenuhi syarat
2	Penjadwalan dan waktu	2	Realistis jadwal dan anggaran	Salah perkiraan (terlalu rendah) waktu dan anggaran pembangunan

3	Sistem fungsionalitas	3	Mengembangkan salah fungsi perangkat lunak	Pengembangan fungsi perangkat lunak yang tidak diperlukan atau salah ditentukan
		4	Mengembangkan pengguna salah Interface	Tidak memadai atau sulit user interface (GUI)
4	Kebutuhan manajemen	5	Gold plating	Penambahan fitur yang tidak perlu ("peluit dan lonceng") kebanggaan karena kepentingan profesional atau tuntutan pengguna.
		6	Melanjutkan aliran perubahan kebutuhan	Yang tidak terkendali dan perubahan tak terduga dalam fungsi sistem dan fitur
5	Subkontrak	7	kekurangan dalam komponen eksternal dilengkapi	Kualitas buruk komponen sistem eksternal disampaikan
		8	kekurangan di eksternal melaksanakan tugas	Miskin kualitas atau pemenuhan terduga tugas eksternal dilakukan
6	Penggunaan sumber daya dan kinerja	9	Real-time kinerja Kekurangan	Miskin kinerja sistem
		10	kemampuan ilmu komputer saring	Ketidakmampuan untuk menerapkan sistem akibat kurangnya solusi teknis dan / atau daya komputasi

6A.2. Kegiatan Pengelolaan Resiko dan Langkah-Langkah

Berbagai kegiatan dan tindakan (biasanya disebut "tindakan manajemen risiko" atau RMAs) dapat diambil. Tujuan RMAs adalah untuk mencegah risiko perangkat lunak, untuk mencapai identifikasi awal item risiko perangkat lunak, dan untuk mengatasinya.

Boehm dan Ross (1989), Boehm (1991), Ropponen dan Lyytinen (2000), dan Karolak (1996), antara lain, telah mengusulkan berbagai tindakan manajemen risiko (lihat Tabel 6A.2).

6A.2 Tabel: Daftar Saran Tindakan terkait Kegiatan Pengelolaan Resiko dan Kontribusinya

No	Software manajemen risiko tindakan (RMA)	Contribution of the RMA		
		Pencegahan	Identifikasi dini SRI	Resolusi SRI
	internal			
1	Penerapan analisis rinci dan menyeluruh dengan persyaratan dan estimasi jadwal dan biaya	x		
2	Efisien proyek organisasi, staf yang memadai dan ukuran tim	x		
3	Pelatihan personil	x		
4	Mengatur untuk dan pelatihan penggantian untuk mengambil alih dalam kasus penjualan dan beban kerja tidak terduga	x		
5	Mengatur partisipasi pengguna dalam proses pembangunan	x		
6	Menerapkan perubahan kontrol yang efisien (perubahan permintaan skrining)	x		
7	Menerapkan langkah-langkah intensif perangkat lunak jaminan mutu seperti pemeriksaan, review desain, dan benchmarking	x		
8	Periodik untuk memeriksa ketersediaan tepat waktu dari perusahaan profesional yang saat ini sibuk dengan proyek lain		x	
9	Mengatur partisipasi anggota staf profesional yang memiliki pengetahuan dan pengalaman dengan SRIS			x
10	Penjadwalan kegiatan SRI terkait sedini mungkin untuk menyediakan waktu luang dalam kasus kesulitan			x

11	Prototyping SRI terkait modul atau aplikasi proyek.			x
12	Menyiapkan skenario untuk modul yang berhubungan SRI rumit atau aplikasi proyek			x
13	Simulasi SRI terkait modul atau aplikasi proyek			x
	Subkontrak RMA			
1	Menyiapkan kontrak komprehensif dan x yang menyeluruh dengan subkontraktor dan pemasok, termasuk review kontrak.	x		
2	Berpartisipasi dalam penyelesaian internal x DNS dan kegiatan perangkat lunak jaminan kualitas subkontraktor untuk dimasukkan dalam kontrak		x	
3	Mengatur untuk "kredit" dari profesional x dengan pengetahuan khusus dan pengalaman jika diperlukan			x
4	Menyewa konsultan untuk mendukung tim x pada tidak adanya pengalaman yang cukup tahu bagaimana dan			x
	Pelanggan RMA			
1	Merumuskan komprehensif dan menyeluruh x kontrak dengan pelanggan, termasuk review kontrak	x		
2	Negosiasi dengan pelanggan untuk x bagian mengubah persyaratan berisiko kembali proyek			x

3	Negosiasi dengan pelanggan untuk mengubah x bagian jadwal ulang proyek berisiko			x

Tindakan-tindakan manajemen risiko dapat dikelompokkan ke dalam kelas-kelas berikut:

- Internal tindakan manajemen risiko, yang diterapkan dalam organisasi perangkat lunak berkembang.
- subkontrak tindakan manajemen risiko, berurusan dengan hubungan antara pengembang perangkat lunak dan subkontraktor dan pemasok.
- tindakan manajemen risiko pelanggan, berurusan dengan hubungan antara pengembang software dan pelanggan.

Implementasi ujung

Dalam RMAs perencanaan, salah satu harus menyadari bahwa:

- Beberapa RMAs dapat mencegah, mengidentifikasi atau menyelesaikan SRIS dari berbagai jenis.
- Beberapa SRIS dapat diobati dengan beberapa RMAs.
- Efisiensi dari RMA bervariasi secara signifikan dengan proyek yang berbeda dan dalam lingkungan yang berbeda.

6A.3. Proses Manajemen Resiko

Proses manajemen risiko menggabungkan kegiatan perencanaan, kegiatan implementasi dan pemantauan. Elaine M. Hall (1998) telah menulis sebuah buku yang didedikasikan terutama untuk proses ini.

Perencanaan kegiatan

Beberapa kegiatan perencanaan bertujuan untuk memulai tindakan-tindakan manajemen risiko yang dapat merespon perangkat lunak risiko diidentifikasi dan dievaluasi sebelumnya. perencanaan kegiatan serupa (walaupun tidak dengan derajat ketelitian yang sama) merupakan bagian dari proses proposal draft review (lihat Bab 5).

Kegiatan perencanaan masing-masing meliputi:

- Identifikasi Item-Item dari Resiko Perangkat Lunak

Alat utama yang mendukung identifikasi SRIS adalah daftar orang-orang yang menentukan tim, proyek dan situasi pelanggan yang mungkin membawa risiko perangkat lunak. Daftar-pembanding jenis ini telah diusulkan oleh Boehm dan Ross (1989), Boehm (1991), Barki et al. (1993), dan Ropponen dan Lyytinen (2000).

Identifikasi risiko item perangkat lunak harus dimulai dengan awal yang sebenarnya dari proyek (pra-proyek tahap) dan diulang secara berkala selama proyek sampai selesai.

- Evaluasi terhadap Identifikasi SRI's

  Evaluasi SRIS diidentifikasi berkaitan terutama dengan:

  - Memperkirakan kemungkinan adanya risiko perangkat lunak akan terwujud jika tidak ada RMA diambil - yaitu, Prob (tikar)
  - Memperkirakan kerusakan dalam hal suatu SRI tidak terwujud - yaitu, Est (bendungan). Perkiraan Prob (tikar) dan Est (bendungan) dapat didasarkan pada pengalaman yang diperoleh dalam proyek-proyek sebelumnya, dengan menggunakan model simulasi, dan sebagainya.

Evaluasi harus diikuti dengan penentuan prioritas tentang SRIS dan resolusi mereka. Harus jelas bahwa SRI menampilkan Prob tinggi (tikar) dan tinggi Est (bendungan) adalah prioritas tinggi dan bahwa SRI menampilkan Prob (tikar) dan rendah rendah Est (bendungan) adalah prioritas rendah.

Salah satu metode yang umum digunakan untuk memprioritaskan SRIS adalah dengan menghitung kerusakan yang diharapkan mereka, disebut "eksposur risiko" - Exp (risiko) - di mana:

$$\text{Exp (resiko)} = \text{Est (bendungan)} \times \text{Prob (tikar)}$$

- Perencanaan RMA

  Adalah tugas tim risiko perangkat lunak untuk mempertimbangkan cara-cara alternatif untuk menyelesaikan SRIS diidentifikasi. RMAs meliputi berbagai internal, subkontraktor dan tindakan pelanggan.

  Tabel 6A.2 menyediakan daftar RMAs mungkin dan kontribusi mereka dengan pencegahan atau resolusi SRIS.

  Dalam penyusunan daftar disarankan RMAs, tim perencanaan harus mempertimbangkan:

  - Prioritas ditugaskan di SRI
  - Hasil yang diharapkan dari sebuah (resolusi lengkap atau sebagian) direncanakan RMA

- Biaya dan upaya organisasi yang diperlukan untuk pelaksanaan RMA.

Pelaksanaan

Pelaksanaan

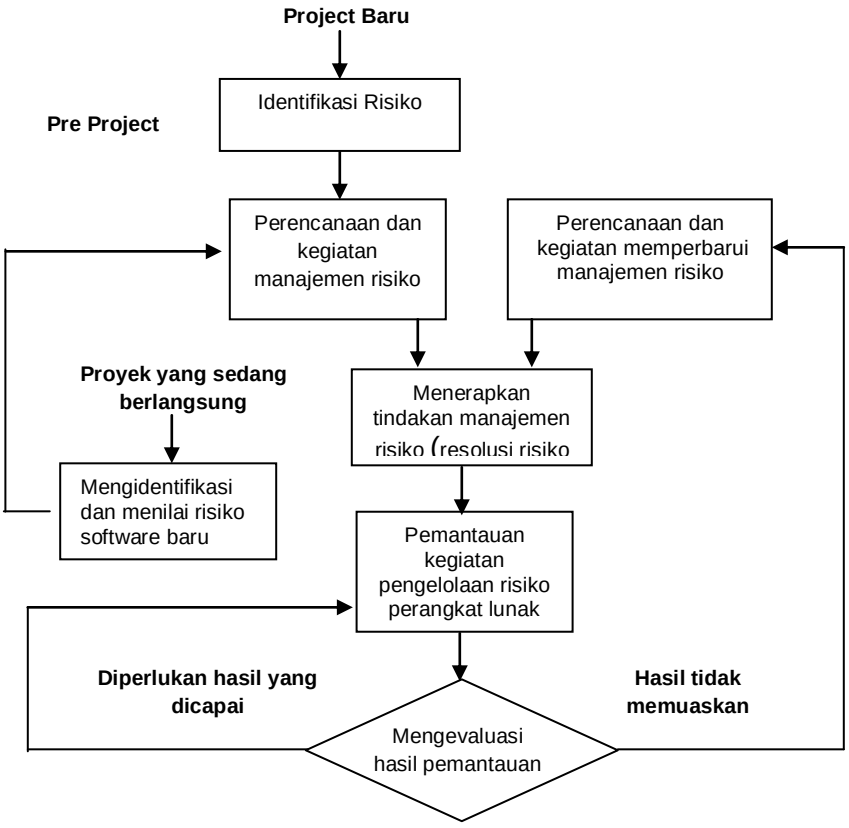
Pelaksanaan rencana manajemen risiko mensyaratkan bahwa anggota staf ditugaskan secara pribadi bertanggung jawab untuk setiap RMA dan jadwal pelaksanaannya.

Pemantauan pelaksanaan rencana manajemen risiko

Sistematis, kegiatan berkala diharuskan untuk memantau pelaksanaan rencana manajemen risiko. Tujuan dari kegiatan pemantauan adalah:

- Tentukan efisiensi RMAs
- Perbarui evaluasi resiko dengan mempertimbangkan SRIS baru diidentifikasi.

Proses perangkat lunak manajemen resiko digambarkan dalam gambar 6A.1 sebagai berikut



:

Gambar 6A.1. Proses Pengelolaan Resiko Perangkat Lunak

Bab7. Menyatukan Kegiatan Kualitas dalam Siklus Hidup Proyek

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menggambarkan ragam model pengembangan software dan mendiskusikan perbedaan diantara mereka
2. Menjelaskan pertimbangan yang mempengaruhi intensitas penerapan kegiatan penjaminan kualitas
3. Menjelaskan perbedaan aspek dari verifikasi, validasi dan kualifikasi yang tergabung dalam kegiatan penjaminan kualitas
4. Menggambarkan model perencanaan SQA yang menghapuskan kecacatan, ketidakefektifan dan biaya
5. Menjelaskan segala kemungkinan penggunaan model tersebut

Bagian pertama dari bab ini adalah yang didedikasikan untuk pengembangan berbagai perangkat lunak model yang digunakan saat ini. Kesepakatan bagian yang tersisa dengan tujuan kegiatan jaminan kualitas perangkat lunak dilakukan selama proyek siklus hidup, integrasi mereka dalam proses pembangunan, dan faktor-faktor yang dianggap sebelum menerapkannya.

Orang mungkin bertanya mengapa tidak dimulai dengan aktivitas SQA dan menghilangkan diskusi model pengembangan perangkat lunak? Pertanyaan ini tidak hanya retoris. Model pengembangan perangkat lunak memberikan satu set konsep yang terkoordinasi dan metodologi yang diperlukan untuk melaksanakan pengembangan perangkat lunak. Seperti seperti itu, mereka termasuk definisi kegiatan utama yang diperlukan untuk pengembangan, urutan yang tepat untuk kinerja mereka, dan pencapaian mereka. Dengan memutuskan apa model yang harus diterapkan, pemimpin proyek menentukan bagaimana proyek akan dilakukan. Sebagian besar kegiatan jaminan kualitas berlangsung di bersamaan dengan penyelesaian atau pemeriksaan tonggak kegiatan, yang memerlukan tinjauan dari kegiatan pengembangan produk sebelumnya selesai. Oleh karena itu, profesional SQA harus berkenalan dengan berbagai model rekayasa perangkat lunak untuk mampu menyiapkan kualitas rencana yang benar terintegrasi ke dalam rencana proyek.

Sisa bagian pertama dari bab dengan faktor-faktor yang mempengaruhi pilihan kegiatan kualitas perangkat lunak yang akan diintegrasikan dalam pengembangan proses. Empat bab berikut (Bab 8 sampai 11) berurusan dengan spesifik metodologi perangkat lunak yang berkualitas untuk diterapkan pada setiap tahap pembangunan panggung dan dalam tahap operasi-pemeliharaan.

Bagian kedua dari bab ini didedikasikan untuk model untuk menilai suatu rencana untuk efektivitas cacat-removal SQA dan biaya. Model, kelipatan model penyaringan, didasarkan pada data yang diperoleh dari survei terhadap asal-usul cacat, persentase penghilangan cacat dicapai oleh berbagai

kegiatan jaminan kualitas, dan biaya cacat-penghapusan terjadi pada berbagai pengembangan fase. Model ini memungkinkan perbandingan kuantitatif jaminan mutu kebijakan yang diwujudkan dalam rencana jaminan kualitas.

7.1. Metodologi Pengembangan Software Klasik dan Yang Lain

Empat model proses pengembangan software yang akan dibicarakan adalah :

- Model Software Development Life Cycle (SDLC)
- Model Prototyping
- Model Spiral
- Model Object Oriented

The Software Development Life Cycle model (model SDLC) adalah model klasik (masih berlaku sekarang), tetapi memberikan yang paling komprehensif deskripsi proses yang tersedia. Model ini menampilkan bangunan utama blok untuk seluruh proses pembangunan, yang digambarkan sebagai sebuah urutan linier. Dalam tahap awal dari proses pengembangan perangkat lunak, dokumen desain produk disusun, dengan versi pertama dari program komputer selesai dan disajikan untuk evaluasi hanya pada cukup tahap akhir dari proses. The Model SDLC dapat berfungsi sebagai kerangka kerja bagi model lain disajikan.

Model prototyping didasarkan pada penggantian satu atau lebih SDLC model tahap oleh proses evolusi, di mana prototipe perangkat lunak digunakan untuk komunikasi antara pengembang dan pengguna dan pelanggan. Prototipe yang disampaikan kepada perwakilan pengguna untuk evaluasi. The pengembang kemudian melanjutkan pengembangan suatu prototipe yang lebih canggih, yang juga disampaikan untuk evaluasi. Proses evolusi terus sampai proyek software selesai atau prototipe perangkat lunak telah mencapai tahap yang diinginkan. Dalam hal ini, seluruh proses pembangunan dapat dilaksanakan sesuai dengan metodologi yang berbeda, misalnya klasik SDLC model.

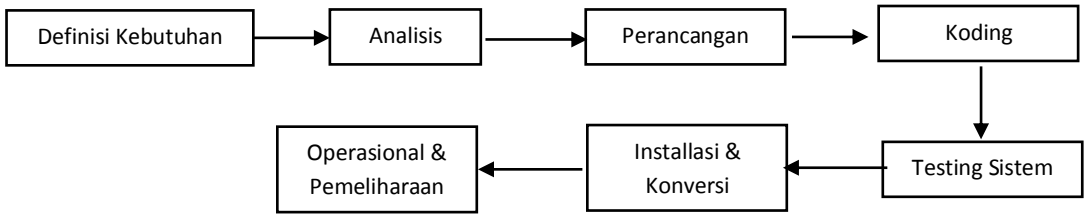
Model spiral menyediakan metodologi untuk menjamin kinerja yang efektif pada setiap fase SDLC model. Ini melibatkan proses yang berulang-ulang yang mengintegrasikan komentar pelanggan dan persyaratan perubahan, analisis resiko dan resolusi, dan perencanaan sistem perangkat lunak dan rekayasa kegiatan. Salah satu atau iterasi lebih dari model spiral mungkin diperlukan untuk menyelesaikan masing-masing proyek fase SDLC. Tugas rekayasa terkait mungkin dilakukan menurut model apapun satu atau kombinasi dari mereka.

Model berorientasi objek menggabungkan kembali besar-besaran perangkat lunak oleh mengintegrasikan modul-modul yang dapat digunakan kembali ke dalam sistem perangkat lunak baru. Dalam kasus dimana tidak modul perangkat lunak dapat digunakan kembali (disebut objek atau komponen) yang tersedia, pengembang dapat melakukan suatu prototipe atau proses SDLC untuk menyelesaikan baru mengembangkan sistem perangkat lunak. Semua empat model akan disajikan secara rinci dalam empat bagian berikutnya. Diskusi rinci dari metodologi masing-masing tersedia dalam

perangkat lunak teknik dan sistem analisis sastra, terutama Pressman (2000) dan Kendall dan Kendall (1999).

7.1.1. Model SDLC

Model ditunjukkan dalam gambar menyajikan tujuh fase proses sebagai berikut :



Gambar 7.1. Model Waterfall (dalam arah mendatar)

Gambar 7.1. menyajikan proses tujuh tahap sebagai berikut :

- Persyaratan definisi. Untuk fungsi sistem perangkat lunak untuk dikembangkan, pelanggan harus menetapkan persyaratan mereka. Di banyak kasus sistem perangkat lunak adalah bagian dari sistem yang lebih besar. Informasi tentang bagian lain dari sistem diperluas membantu kerjasama antara tim dan mengembangkan antarmuka komponen.
- Analisis. Upaya utama di sini adalah untuk menganalisis implikasi persyaratan ' untuk membentuk model perangkat lunak sistem awal.
- Desain. Tahap ini melibatkan definisi rinci dari output, input dan pengolahan prosedur, termasuk struktur data dan database, perangkat lunak struktur, dll
- Coding. Pada tahap ini, desain diterjemahkan ke dalam kode. Coding melibatkan kegiatan jaminan kualitas seperti inspeksi, tes unit dan integrasi tes.
- Sistem tes. Sistem tes dilakukan setelah tahap pengkodean selesai. Tujuan utama pengujian adalah untuk mengungkap sebagai kesalahan perangkat lunak sebanyak mungkin sehingga mencapai tingkat yang dapat diterima dari kualitas perangkat lunak sekali koreksi telah selesai. Sistem tes dilakukan oleh pengembang perangkat lunak sebelum perangkat lunak diberikan kepada pelanggan. Dalam banyak kasus pelanggan melakukan tes perangkat lunak independen ("penerimaan tes ") untuk meyakinkan dirinya sendiri bahwa pengembang

telah memenuhi semua komitmen dan bahwa tidak ada reaksi yang tidak diantisipasi atau perangkat lunak yang salah diantisipasi. Hal ini sangat umum bagi pelanggan untuk meminta pengembang untuk bergabung dengan dia dalam melakukan tes sistem sambungan, prosedur yang menyelamatkan waktu dan sumber daya yang dibutuhkan untuk tes penerimaan terpisah.

- Instalasi dan konversi. Setelah sistem perangkat lunak yang disetujui, sistem terinstal untuk melayani sebagai firmware, yaitu, sebagai bagian dari informasi sistem yang merupakan komponen utama dari sistem diperluas. Jika sistem informasi baru untuk menggantikan sistem yang ada, sebuah software konversi proses harus dimulai untuk memastikan bahwa organisasi kegiatan tersebut berjalan tanpa gangguan selama tahap konversi.
- Reguler operasi dan pemeliharaan. Operasi software Reguler dimulai sekali instalasi dan konversi telah selesai. Sepanjang operasi normal periode, yang biasanya berlangsung selama beberapa tahun atau sampai generasi perangkat lunak baru muncul di tempat kejadian, pemeliharaan diperlukan. Pemeliharaan menggabungkan tiga jenis layanan: perbaikan – perbaikan perangkat lunak kesalahan diidentifikasi oleh pengguna selama operasi; adaptif – menggunakan fitur-fitur software yang ada untuk memenuhi persyaratan baru; dan perfektif - menambahkan fitur kecil baru untuk meningkatkan kinerja perangkat lunak.

Jumlah fase dapat bervariasi sesuai dengan karakteristik proyek. Dalam kompleks, model berskala besar, beberapa tahap dibagi, menyebabkan mereka nomor untuk tumbuh sampai delapan, sembilan atau lebih. Dalam proyek-proyek yang lebih kecil, mungkin beberapa tahap terserap, mengurangi jumlah tahapan untuk enam, lima atau bahkan empat fase. Pada akhir setiap fase, keluaran diperiksa dan dievaluasi oleh pengembang dan, dalam banyak kasus, oleh pelanggan juga. Kemungkinan hasil penelaahan dan evaluasi meliputi:

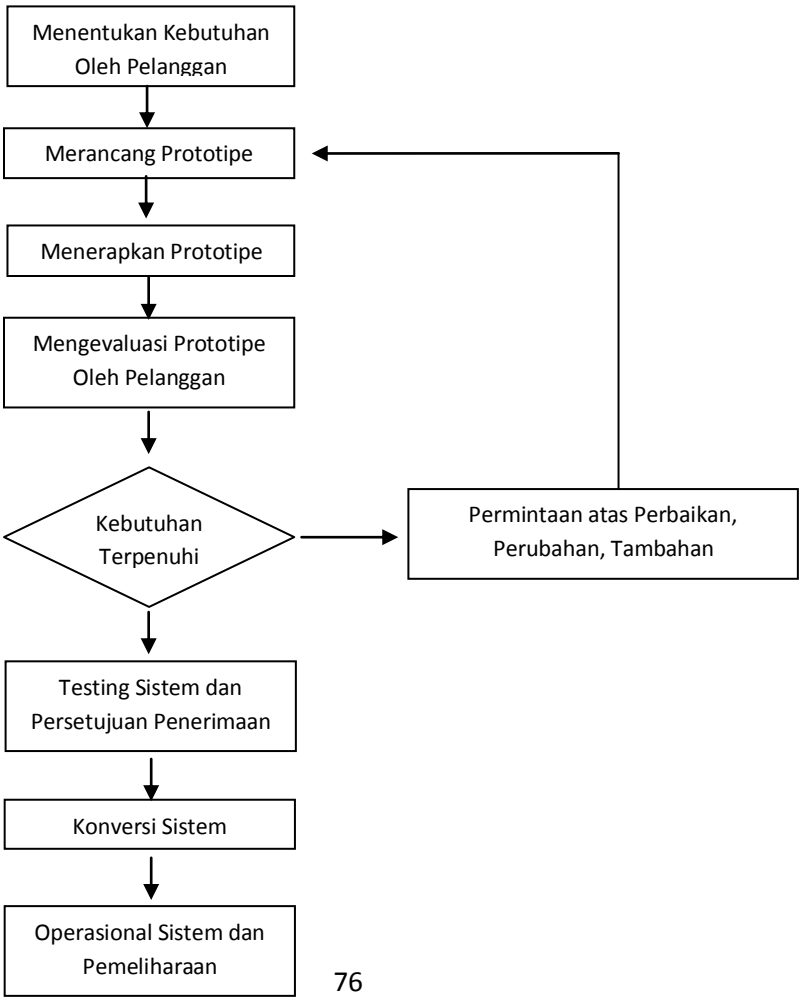
- Persetujuan output fasa dan melanjutkan ke tahap berikutnya, atau
- Tuntutan untuk memperbaiki, mengulang atau mengubah bagian dari fase terakhir, dalam tertentu kasus, kembali ke fase awal yang diperlukan. Lebar garis yang menghubungkan kotak persegi panjang dalam ilustrasi mencerminkan probabilitas relatif dari hasil yang berbeda. Jadi, yang paling umum proses yang dilakukan adalah urutan linear (tidak ada atau hanya koreksi kecil). Kita harus dicatat, bagaimanapun, bahwa model tersebut menekankan langsung pengembangan kegiatan dan tidak menunjukkan saham pelanggan dalam pengembangan proses. Model air terjun klasik

disarankan oleh Royce (1970) dan kemudian disajikan dalam bentuk yang umum dikenal oleh Boehm (1981). Hal ini menyediakan fondasi bagi mayoritas standar jaminan kualitas perangkat lunak utama bekerja, seperti IEEE Std 1012 (IEEE, 1998) dan IEEE Std 12207 (IEEE, 1996, 1997a, 1997b), untuk menyebutkan hanya dua.

7.1.2. Model Prototipe

Metodologi prototyping melakukan penggunaan :

- Dikembangkan menggunakan teknologi informasi, penamaan, penggunaan aplikasi tingkat lanjut yang memperkenankan kita menghasilkan program dengan cepat dan pengembangan yang mudah terhadap prototipe software
- Dikombinasikan dengan partisipan yang aktif dalam proses pengembangan oleh pelanggan dan dan user yang mampu dalam memeriksa dan mengevaluasi prototipe.



Gambar 7.2. Model Prototyping

Keuntungan prototyping :

- Proses pengembangan lebih cepat
- Berperan penting dalam menghemat sumber daya pengembangan (orang /hari)
- Lebih baik dalam memenuhi kebutuhan dan mengurangi resiko terhadap kegagalan proyek
- Lebih mudah dan cepat dalam memahami sistem baru

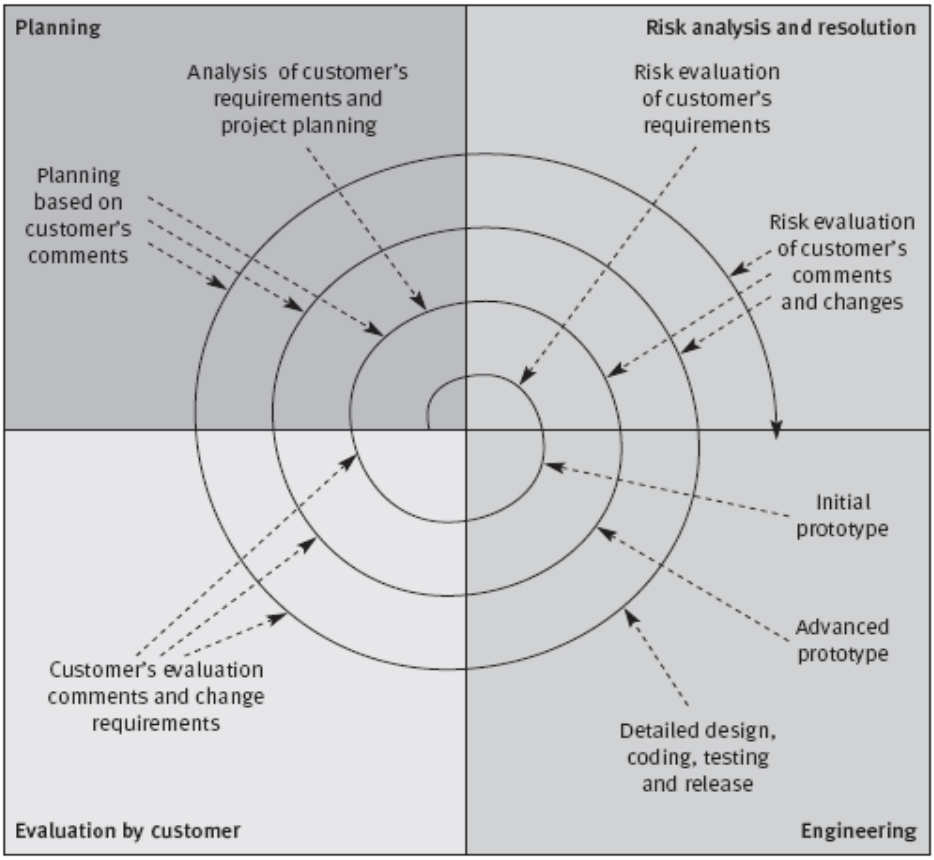
Kerugian prototyping :

- Berkurangnya fleksibilitas dan adaptasi terhadap perubahan dan penambahan
- Berkurangnya persiapan terhadap kejadian tidak terduga dari sebuah kesalahan

7.1.3. Model Spiral

Model spiral, sebagaimana telah diubah oleh Boehm (1988, 1998), menawarkan perbaikan metodologi untuk mengawasi proyek-proyek pembangunan besar dan lebih kompleks menampilkan prospek yang lebih tinggi untuk kegagalan, khas banyak proyek dimulai di dua dekade terakhir. Ini menggabungkan model iteratif yang memperkenalkan dan menekankan analisis risiko dan partisipasi pelanggan menjadi elemen utama SDLC dan metodologi prototyping. Berdasarkan model spiral, pengembangan software dirasa sebagai sebuah proses iterasi, dimana masing-masing iterasi mengikuti kegiatan sebagai berikut :

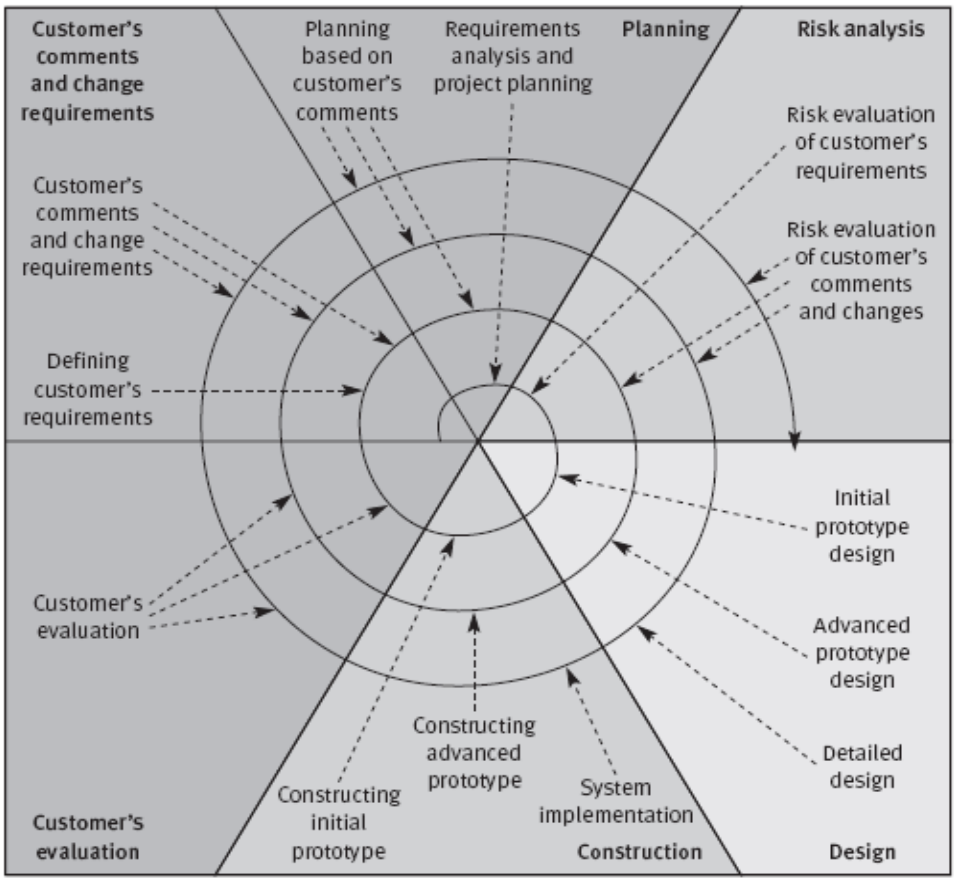
- Perencanaan
- Analisa resiko dan pemecahan ulang masalah
- Aktifitas keteknikan berdasarkan tahapan proyek : perancangan, koding, testing, installasi dan mengeluarkan produk.
- Evaluasi pelanggan termasuk komentar, perubahan dan penambahan kebutuhan.



Gambar 7.3. Model Spiral (Boehm, 1988)

Model maju spiral, yang Win-Win Spiral Model (Boehm, 1998), meningkatkan model Spiral (Boehm, 1988) lebih jauh lagi. Model canggih tempat ekstra penekanan pada komunikasi dan negosiasi antara pelanggan dan pengembang. Nama model merujuk pada fakta bahwa dengan menggunakan proses, pelanggan "menang" dalam bentuk kesempatan diperbaiki untuk menerima sistem yang paling memuaskan kebutuhan, dan pengembang "menang" dalam bentuk meningkatkan kesempatan untuk tinggal di dalam anggaran dan menyelesaikan proyek oleh setuju tanggal. Hal ini dicapai dengan meningkatnya penekanan pada partisipasi pelanggan dan kegiatan rekayasa. Revisi ini dalam proses pembangunan ditunjukkan secara grafik oleh dua bagian dari spiral didedikasikan untuk partisipasi pelanggan: yang pertama berkaitan dengan evaluasi pelanggan dan yang kedua dengan pelanggan komentar dan persyaratan perubahan. Rekayasa aktivitas juga ditunjukkan dalam dua bagian dari spiral: yang pertama adalah didedikasikan untuk desain dan yang kedua untuk konstruksi.

Dengan mengevaluasi kemajuan proyek pada akhir masing-masing bagian, pengembang dapat lebih baik mengendalikan seluruh proses pembangunan. Dengan demikian, dalam model spiral maju, ditunjukkan dalam Gambar 7.4, berikut enam kegiatan dilakukan dalam setiap iterasi:



Gambar 7.4. Model Spiral Lanjut

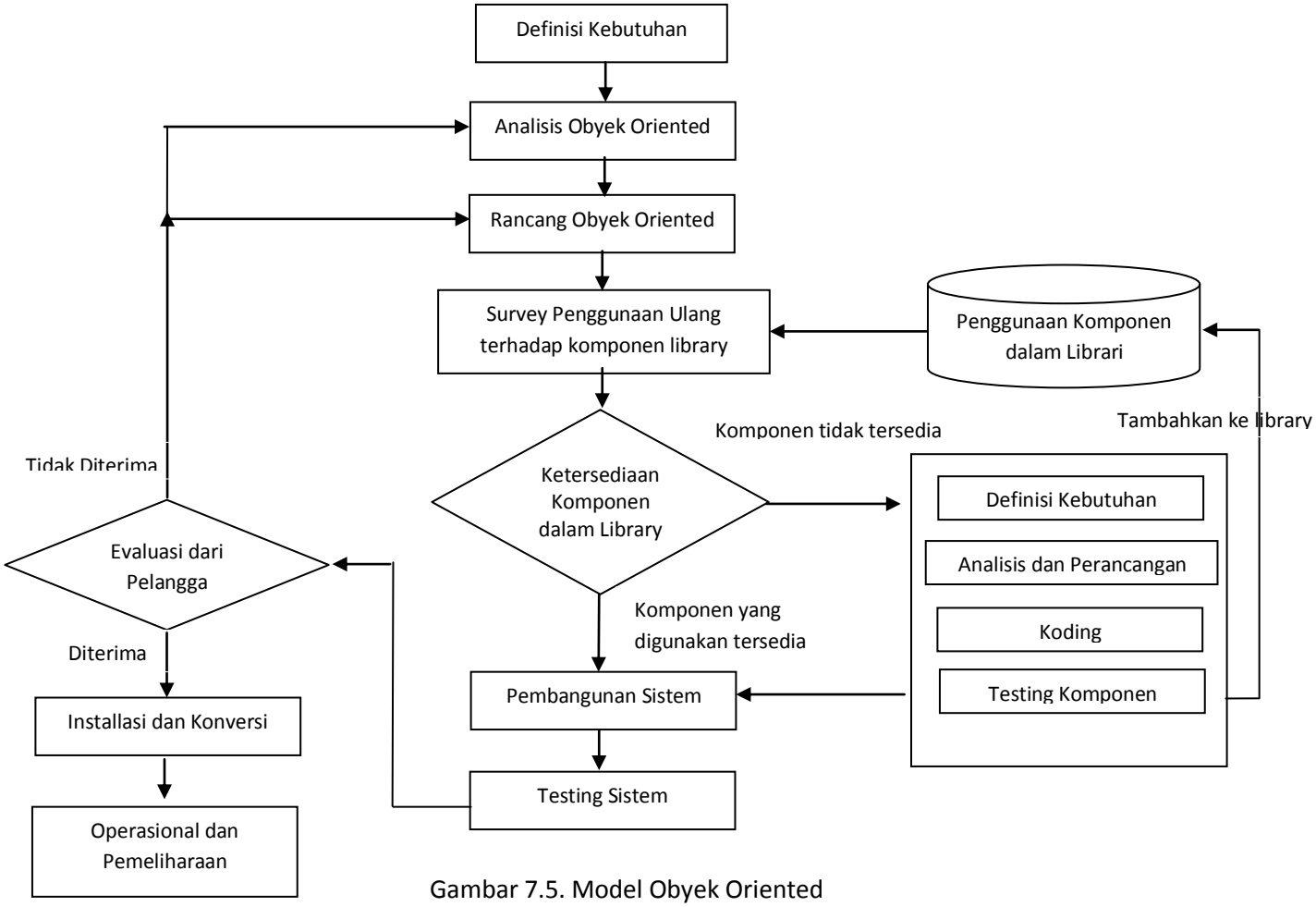
- Nasabah spesifikasi kebutuhan, komentar dan tuntutan perubahan
- Developer kegiatan perencanaan
- Developer analisis risiko dan resolusi
- Developer desain kegiatan
- konstruksi Pengembang kegiatan-kegiatan yang berhubungan dengan coding, pengujian, instalasi dan lepaskan
- Nasabah evaluasi.

7.1.4. Model Obyek Oriented

Model berorientasi objek berbeda dari model lain dengan intensif penggunaan kembali komponen perangkat lunak. Metodologi ini ditandai dengan mudah integrasi modul perangkat lunak yang ada (disebut objek atau komponen) ke baru dikembangkan sistem perangkat lunak. Sebuah perpustakaan komponen perangkat lunak melayani ini Tujuan dengan menyediakan komponen perangkat lunak untuk digunakan kembali.

Jadi, menurut model berorientasi objek seperti yang ditunjukkan pada Gambar 7.5, proses pembangunan dimulai dengan urutan analisa berorientasi obyek dan desain. Tahap desain diikuti pembelian komponen yang cocok dari perpustakaan perangkat lunak dapat digunakan kembali, bila tersedia. "Reguler" pembangunan dilakukan sebaliknya. Salinan dari komponen software yang baru dikembangkan kemudian "ditebar" di perpustakaan perangkat lunak untuk digunakan kembali di masa depan. Diharapkan bahwa saham komponen perangkat lunak tumbuh di perpustakaan perangkat lunak dapat digunakan kembali akan memungkinkan penggunaan kembali substansial dan meningkatkan perangkat lunak, sebuah tren yang akan memungkinkan mengambil keuntungan yang lebih besar dari sumber daya sebagai berikut:

- ✓ Ekonomi - Biaya mengintegrasikan komponen perangkat lunak dapat digunakan kembali adalah jauh lebih rendah dari biaya pengembangan komponen baru.
- ✓ Peningkatan kualitas - komponen software yang digunakan adalah diduga mengandung jauh lebih sedikit cacat dari komponen software yang baru dikembangkan karena deteksi kesalahan oleh pengguna mantan.



Gambar 7.5. Model Obyek Oriented

- ✓ Waktu Pengembangan lebih pendek– Pengintegrasian komponen perangkat lunak bisa kembali mengurangi skeduling tekanan. Seperti itu, keuntungan metodologi yang berorientasi, metodologi akan digunakan perangkat lunak.

7.2. Faktor yang sangat berpengaruh terhadap kegiatan penjaminan kualitas dalam proses pengembangan.

Siklus hidup proyek kegiatan penjaminan mutu adalah proses yang berorientasi, dengan kata lain, terkait dengan penyelesaian fase proyek, penyelesaian dari sebuah kejadian penting proyek, dan sebagainya.

Kegiatan jaminan kualitas akan diintegrasikan ke dalam rencana pembangunan yang mengimplementasikan satu atau lebih model pengembangan perangkat lunak - air terjun, prototyping, spiral, berorientasi objek atau model.

Perencana jaminan kualitas dalam sebuah proyek diperlukan untuk menentukan :

- Daftar jenis kegiatan jaminan kualitas yang diperlukan dalam sebuah proyek
- Kegiatan untuk masing-masing jaminan kualitas :
    - Waktu
    - Tipe dari kegiatan jaminan kualitas yang diterapkan
    - Siapa yang melakukan kegiatan dan sumber daya yang diperlukan
    - Sumber daya yang diperlukan untuk menghilangkan kecacatan dan mengenalkan perubahan.

Faktor-faktor yang mempengaruhi intensitas kegiatan yang diperlukan dalam menjamin kualitas perangkat lunak terbagai atas :

1) Faktor Proyek
    - Besaran proyek
    - Teknis kompleksitas dan kesulitan
    - Tingkat dapat digunakan kembalinya komponen perangkat lunak
    - Resiko hasil kegagalan jika proyek gagal
2) Faktor Tim
    - Kualifikasi Profesional dari anggota tim
    - Pengetahuan Tim dengan proyek dan pengalamannya di daerah
    - Ketersediaan staf anggota yang dapat dengan profesional mendukung tim
    - Keakraban dengan anggota tim, dengan kata lain persentase staf anggota baru dalam tim

Contoh 1 :

Sebuah tim pengembangan perangkat lunak telah merencanakan kegiatan jaminan kualitas untuk konsumen baru proyek klub. Kontrak proyek ini, ditandatangani oleh sebuah toko furnitur terkemuka, adalah 11 konsumen tim menangani proyek klub dalam tiga tahun terakhir. Tim ini memperkirakan bahwa sekitar tujuh orang-bulan perlu diinvestasikan oleh dua anggota tim yang ditugaskan untuk

proyek, yang durasinya diperkirakan empat bulan. Diperkirakan bahwa komponen perpustakaan yang dapat digunakan kembali dapat memberi 90% dari proyek perangkat lunak. Tiga kegiatan jaminan kualitas telah direncanakan oleh pemimpin proyek. Kegiatan jaminan kualitas dan durasi mereka tercantum dalam Tabel 7.1.

Tabel 7.1. : Jangka waktu kegiatan jaminan kualitas - contoh klub konsumen

No	kegiatan Jaminan kualitas	Jangka waktu kegiatan jaminan mutu (hari)	Durasi koreksi dan perubahan (hari)
1	Desain pemeriksaan dari definisi persyaratan	0,5	1
2	Pemeriksaan dari desain	1	1
3	uji Sistem dari selesainya paket perangkat lunak	4	2

Sebuah tim pengembangan perangkat lunak telah merencanakan kegiatan jaminan kualitas untuk konsumen baru proyek klub. Kontrak proyek ini, ditandatangani oleh sebuah toko furnitur terkemuka, adalah 11 konsumen tim menangani proyek klub dalam tiga tahun terakhir. Tim ini memperkirakan bahwa sekitar tujuh orang-bulan perlu diinvestasikan oleh dua anggota tim yang ditugaskan untuk proyek, yang durasinya diperkirakan empat bulan. Diperkirakan bahwa komponen perpustakaan yang dapat digunakan kembali dapat memberi 90% dari proyek perangkat lunak. Tiga kegiatan jaminan kualitas telah direncanakan oleh pemimpin pPertimbangan utama yang mempengaruhi program ini adalah:

- Tingkat pengetahuan tim dengan subjek
- persentase tinggi dari penggunaan kembali perangkat lunak
- Ukuran proyek (dalam hal ini, menengah)
- Resiko hasil kegagalan jika proyek gagal.

Contoh 2

unit pengembangan Perangkat lunak real-time dari departemen sistem informasi sebuah rumah sakit telah ditugaskan untuk mengembangkan kecanggihan suatu sistem pemantauan pasien. Unit pemantauan baru ini untuk menggabungkan unit kamar pasien dengan unit kontrol. Unit kamar pasien dimaksudkan untuk menghubungkan dengan beberapa jenis peralatan medis, yang disediakan oleh berbagai pabrik, yang mengukur berbagai indikator kondisi pasien. Sebuah unit kontrol canggih akan ditempatkan di stasiun perawat, dengan data yang akan dikomunikasikan kepada unit seluler yang dibawa oleh dokter.

Pemimpin proyek memperkirakan bahwa akan diperlukan 14 bulan untuk melengkapi sistem; tim lima akan dibutuhkan, dengan investasi dari total 40 orang-bulan. Dia memperkirakan bahwa hanya 15% dari komponen dapat diperoleh dari perpustakaan komponen yang dapat digunakan kembali. Metodologi SDLC dipilih untuk mengintegrasikan aplikasi dua prototip dari unit kamar pasien dan dua prototipe dari unit kontrol untuk tujuan memperbaiki komunikasi dengan pengguna dan meningkatkan timbal balik dari komentar pada tahap analisis dan desain.

Pertimbangan utama yang mempengaruhi program ini adalah:

- Tinggi kompleksitas dan kesulitan sistem
- Rendahnya persentase perangkat lunak yang dapat digunakan kembali tersedia
- Besarnya ukuran proyek
- Resiko Tinggi hasil kegagalan jika proyek gagal.

7.3. Verifikasi, Validasi dan Kualifikasi

- **Verifikasi** : proses mengevaluasi sebuah sistem atau komponen untuk menentukan apakah produk dari sebuah fase pengembangan memuaskan dibandingkan kondisi yang dibebankan pada awal fase.
- **Validasi** : proses mengevaluasi sebuah sistem atau komponen selama atau sampai akhir dari proses pengembangan untuk menentukan apakah hal ini memenuhi persyaratan khusus.
- **Kualifikasi** : proses yang digunakan untuk menentukan apakah sebuah sistem atau komponen cocok untuk opersional yang digunakan.

Tabel 7.2. Durasi kegiatan jaminan kualitas – contoh sistem monitoring pasien

No	Kegiatan Jaminan Kualitas	Durasi kegiatan (hari)	Durasi perbaikan dan perubahan (hari)
1	peninjauan Desain definisi persyaratan	2	1
2	peninjauan Desain analisis unit kamar pasien	2	2
3	peninjauan Desain analisis unit kontrol	1	2
4	peninjauan Desain rancangan awal	1	1
5	Pemeriksaan desain unit kamar pasien	1	2
6	Pemeriksaan desain unit kontrol	1	3
7	peninjauan Desain prototipe unit kamar pasien	1	1
8	peninjauan Desain prototipe unit kontrol	1	1
9	Pemeriksaan rincian desain untuk setiap perangkat lunak yang berhubungan dengan komponen	3	3
10	peninjauan Desain rencana uji untuk unit kamar pasien dan unit	3	1

	control		
11	pengujian Unit kode software untuk setiap modul yang berhubungan dengan unit kamar pasien	4	2
12	Integrasi uji kode software unit kamar pasien	3	3
13	Integrasi uji kode software unit kontrol	2	3
14	pengujian selesainya sistem perangkat lunak	10	5
15	peninjauan Desain petunjuk pengguna	3	2

Menurut definisi IEEE, verifikasi memeriksa konsistensi dari produk yang dikembangkan dengan produk yang dikembangkan di tahap sebelumnya. Ketika melakukan demikian, pemeriksa mengikuti proses pengembangan dan mengasumsikan bahwa semua fase pengembangan terdahulu telah selesai dengan benar, seperti pada awal direncanakan atau setelah penghapusan semua kerusakan ditemukan. Asumsi ini memaksa pemeriksa untuk mengabaikan penyimpangan dari persyaratan asli pelanggan yang mungkin telah diperkenalkan selama proses pengembangan.

Validasi mewakili kepentingan pelanggan dengan memeriksa tingkat pemenuhan terhadap persyaratan asli nya. Ulasan validasi Komprehensif cenderung untuk meningkatkan kepuasan pelanggan dari sistem.

Kualifikasi berfokus pada aspek operasional, di mana pemeliharaan adalah persoalan utama. Sebuah komponen perangkat lunak yang telah dikembangkan dan didokumentasikan sesuai dengan standar profesional dan gaya dan struktur prosedur konvensi diharapkan akan lebih mudah untuk dipelihara daripada memberikan improvisasi pengkodingan yang bagus namun tidak mengikuti gaya prosedur pengkodingan yang dikenal dan sebagainya.

Perencana diperlukan untuk menentukan aspek-aspek ini yang harus diperiksa dalam setiap kegiatan jaminan kualitas.

7.4. Model SQA untuk menghilangkan kecacatan secara efektif dan mengurangi biaya.

Transaksi model dengan dua aspek kuantitatif dari rencana SQA terdiri dari beberapa kegiatan pendeteksian kerusakan:

1) Total efektivitas rencana dalam menghilangkan cacat proyek.
2) Biaya total penghilangan cacat proyek

Rencana itu sendiri adalah untuk diintegrasikan dalam proses pengembangan suatu proyek.

7.4.1. Data

Penerapan model ini didasarkan pada tiga jenis data, dijelaskan di bawah judul berikut.

Kerusakan asal distribusi

asal Kerusakan (tahap di mana kerusakan diperkenalkan) didistribusikan seluruhnya ke proses pengembangan, dari inisiasi proyek untuk penyelesaiannya. Survei yang dilakukan oleh pengembang perangkat lunak besar, seperti IBM dan TRW, diringkas oleh Boehm (1981, Bab 24) dan Jones (1996, Bab 3), menunjukkan pola yang serupa distribusi kerusakan. pembangun perangkat lunak profesional percaya bahwa pola ini tidak berubah secara hakikat dalam dua dekade terakhir. Distribusi karakteristik asal kerusakan perangkat lunak, berdasarkan Boehm (1981) dan Jones (1996), akan ditampilkan dalam Tabel 7.3.

Tabel 7.3: Sebuah distribusi karakteristik asal kerusakan perangkat lunak

No	Tahap pengembangan Software	Rata-rata persentase asal kerusakan dari fase
1	spesifikasi Persyaratan	15%
2	Desain	35%
3	Coding (pengkodean 30%, integrasi 10%)	40%
4	Dokumentasi	10%

Efektivitas Penghapusan Kesalahan

Diasumsikan bahwa setiap kegiatan jaminan kualitas filter (layar) persentase pasti dari kesalahan yang ada. Perlu dicatat bahwa dalam kebanyakan kasus, persentase kesalahan dihapus agak lebih rendah dari persentase kesalahan yang dideteksi sebagai beberapa perbaikan (sekitar 10% menurut Jones, 1996) adalah tidak efektif atau tidak memadai. Kesalahan yang tersisa, yang tidak terdeteksi dan tidak dikoreksi, akan dilewatkan ke fase pembangunan berturut-turut. Kegiatan jaminan mutu selanjutnya diterapkan terhadap kombinasi kesalahan: yang tersisa setelah kegiatan jaminan kualitas sebelumnya bersama-sama dengan kesalahan "baru", dibuat dalam tahap pengembangan saat ini. Diasumsikan bahwa efektivitas penyaringan kerusakan akumulasi setiap aktivitas jaminan kualitas tidak kurang dari 40% (yaitu, aktivitas menghapus sedikitnya 40% dari kesalahan masuk). tingkat Khas rata-rata efektivitas penyaringan kesalahan untuk berbagai kegiatan jaminan kualitas, dengan tahap pengembangan, berdasarkan Boehm (, 1981 Bab 24) dan Jones (1996, Bab 3 dan 5), tercantum pada Tabel 7.4.

Biaya penghilangan cacat

Data yang dikumpulkan tentang pengembangan biaya proyek menunjukkan bahwa biaya penghilangan cacat dideteksi bervariasi menurut fase pengembangan, sementara kenaikan biaya pada hakikatnya sebagai hasil proses pengembangan. Sebagai contoh, penghapusan cacat desain terdeteksi dalam tahap desain mungkin memerlukan investasi 2,5 hari kerja; penghapusan cacat yang sama mungkin memerlukan 40 hari kerja selama tes penerimaan. Beberapa survei yang dilakukan oleh IBM, TRW, GTE, Boehm dan lain-lain, diringkas oleh Boehm (1981, Bab 4), memperkirakan biaya relatif mengoreksi kesalahan pada setiap tahap pengembangan. Estimasi efektivitas alat jaminan kualitas perangkat lunak dan biaya relatif penghilangan cacat disediakan oleh Boehm dan Basili (2001). Meskipun data penghapusan cacat cukup langka, profesional setuju bahwa biaya proporsional penghilangan cacat tetap konstan sejak survei dilakukan pada 1970-an dan 1980-an. perwakilan rata-rata biaya relatif penghapusan cacat, berdasarkan Boehm (1981) dan Pressman (2000, Bab 8), disajikan pada Tabel 7.5.

Tabel 7.4: Rata-rata penyaringan (penghilangan cacat) efektifitas dari aktivitas jaminan kualitas

No	kegiatan Jaminan kualitas	Rata-rata tingkat efektivitas penyaringan cacat
1	tinjauan spesifikasi Persyaratan	50%
2	pemeriksaan Desain	60%
3	tinjauan Desain	50%
4	pemeriksaan Kode	65%
5	test Unit	50%
6	test Unit setelah pemeriksaan kode	30%
7	uji Integrasi	50%
8	tes Sistem / tes penerimaan	50%
9	tinjauan Dokumentasi	50%

Tabel 7.5: Perwakilan rata-rata biaya relatif penghapusan cacat

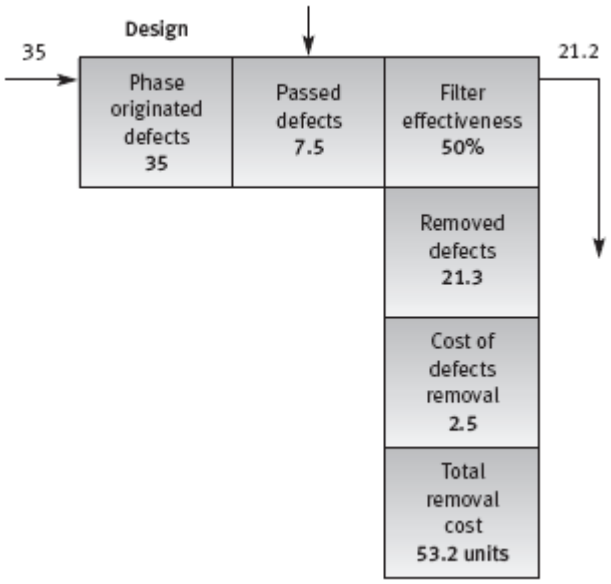
No	Tahap pengembangan perangkat lunak	Rata-rata biaya relatif cacat (unit cost)
1	Spesifikasi persyaratan	1
2	Desain	2.5
3	Tes unit	6,5
4	tes Integrasi	16
5	uji Sistem / tes penerimaan / tinjauan dokumentasi sistem	40
6	Operasi oleh pelanggan (setelah rilis)	110

7.4.2. Model

Model ini didasarkan pada asumsi sebagai berikut:

- Proses pengembangan bersifat linear dan sekuensial, mengikuti model air terjun.
- Sejumlah cacat "baru" diperkenalkan dalam setiap tahap pengembangan. Untuk distribusi mereka, lihat Tabel 7.3.
- Review dan kegiatan uji jaminan kualitas perangkat lunak berfungsi sebagai filter, menghilangkan persentase cacat masuk dan membiarkan sisanya lolos ke tahap pengembangan selanjutnya. Sebagai contoh, jika jumlah cacat yang masuk adalah 30, dan efisiensi penyaringan adalah 60%, maka 18 cacat akan dihapus, sementara 12 cacat akan tetap dan lulus untuk dideteksi oleh aktivitas jaminan kualitas berikutnya. Tipe tingkat efektivitas penyaringan untuk berbagai kegiatan jaminan kualitas ditunjukkan pada Tabel 7.4.
- Pada setiap tahap, cacat masuk adalah jumlah cacat tidak dihapus oleh aktivitas jaminan kualitas terdahulu bersama dengan cacat "baru" diperkenalkan (diciptakan) dalam tahap pengembangan saat ini.
- Biaya penghilangan cacat dihitung untuk setiap kegiatan jaminan kualitas dengan mengalikan jumlah cacat yang dihilangkan dengan biaya relatif menghapus cacat (lihat Tabel 7.5).
- Cacat yang tersisa, sayangnya diserahkan kepada pelanggan, akan terdeteksi oleh dia. Dalam keadaan ini, penghapusan penuh memerlukan biaya penghapusan cacat terberat.

Dalam model, masing-masing kegiatan jaminan kualitas diwakili oleh unit penyaring, seperti yang ditunjukkan Desain pada Gambar 7.6.



Gambar 7.6: Sebuah unit filter untuk efektivitas penghapusan cacat: contoh

Model ini menyajikan kuantitas berikut:

- POD = Phase Originated Defects (from Table 7.3)
- PD = Passed Defects (from former phase or former quality assurance activity)
- %FE = % of Filtering Effectiveness (also termed % screening effectiveness) (from Table 7.4)
- RD = Removed Defects
- CDR = Cost of Defect Removal (from Table 7.5)
- TRC = Total Removal Cost: TRC = RD □ □ CDR.

Ilustrasi pertama berlaku untuk model rencana jaminan kualitas standar ("sistem penyaringan cacat standar ") yang terdiri dari enam kegiatan jaminan mutu (enam filter), seperti ditunjukkan pada Tabel 7.6.

N0	kegiatan Jaminan kualitas	efektivitas penghapusan Cacat	Biaya penghapusan cacat yang terdeteksi (unit cost)
1	review spesifikasi Kebutuhan	50%	1
2	review Desain	50%	2,5
3	Uji Unit - kode	50%	6,5
4	uji Integrasi	50%	16
5	review Dokumentasi	50%	16
6	uji Sistem	50%	40
7	tahap Operasi	100%	110

7.5. Ringkasan

1) Gambarkan macam-macam model pengembangan software dan diskusikan perbedaan diantara mereka.
2) Jelaskan pertimbangan yang mempengaruhi aplikasi terhadap kegiatan jaminan kualitas
3) Jelaskan aspek yang berbeda dari verifikasi, validasi dan kualifikasi dalam sebuah kegiatan jaminan kualitas.
4) Jelaskan model untuk SQA mengurangi kecacatan, ketidakefektifan dan biaya
5) Jelaskan kemungkinan penggunaan model tersebut

Bab8. Review (Peninjauan / Pemeriksaan)

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan tujuan langsung dan tidak langsung dari metodologi review
2. Menjelaskan kontribusi para ahli dari luar organisasi dalam rangka unjuk performansi tugas review
3. Membandingkan tiga metodologi utama dari review

8.1. Tujuan Review

Beberapa tujuan memotivasi review. review tujuan langsung berhubnngan dengan proyek yang sekarang, sedangkan tujuan tidak langsung, lebih bersifat umum, berurusan dengan kontribusi meninjau tepat untuk promosi tim anggota profesional pengetahuan dan peningkatan pembangunan metodologi yang diterapkan oleh organisasi. Tujuan langsung dari review :

- Melacak kesalahan analisis dan perancangan dan bagaimana perbaikannya, perubahan dan penambahan diperlukan dengan mematuhi spesifikasi awal dan perubahan yang disetujui.
- Mengidentifikasi resiko baru yang berakibat terhadap kelengkapan suatu proyek.
- Menemukan penyimpangan terhadap template dan model prosedur dan persetujuan. Perbaikan atas penyimpangan ini diharapkan berkontribusi untuk meningkatkan komunikasi dan koordinasi dalam menghasilkan penyeragaman yang lebih baik terhadap model dokumentasi dan metodenya.
- Menyutujui produk analisis dan perancangan. Persetujuan ini memberikan ijin bagi tim untuk melanjutkan ke fase pengembangan berikutnya.

Tujuan tidak langsung dari review :

- Menyediakan tempat pertemuan informal dalam rangka pertukaran pengetahuan tentang metode pengembangan, tool dan teknik yang digunakan.
- Untuk mencatat kesalahan analisis dan perancangan yang menjadi dasar bagi tindakan perbaikan di masa mendatang. Tindakan perbaikan diharapkan meningkatkan metode pengembangan dengan cara peningkatan keefektifan dan kualitas diantara feature produk lain.

Metode berbagai review berbeda dalam penekanan yang melekat pada berbagai tujuan dan di tingkat keberhasilan dicapai untuk tujuan masing-masing. Oleh karena itu, untuk lebih baik "menyaring"

kesalahan dan dampak jangka panjang yang lebih besar, ganda atau bahkan triple "bersih", dibangun dari antara berbagai Review metode yang tersedia, harus diterapkan.

Review bukan kegiatan yang akan dilakukan sembarangan. Order procedural dan teamwork terletak di jantung Review desain formal, inspeksi dan walkthrough. Setiap peserta diharapkan untuk menekankan daerah nya dari tanggung jawab atau spesialisasi ketika membuat komentar. Pada setiap sesi review, satu individu diberikan tugas inscribing disepakati bersama komentar. Daftar item berikutnya harus mencakup rincian lengkap lokasi cacat dan deskripsi, didokumentasikan dengan cara yang nantinya akan memungkinkan pengambilan penuh oleh tim pengembangan. Namun, karena kecenderungan manusia untuk mencoba merancang solusi di tempat dan, sering, untuk ngelantur masalah tangensial atau, bahkan lebih buruk lagi, untuk hal-hal pribadi selama rapat, seorang coordinator diperlukan untuk mempertahankan kontrol diskusi dan tetap di jalur.

Secara umum, pengetahuan bahwa produk analisis atau desain akan ditinjau merangsang tim pengembangan untuk bekerja lebih maksimal. Ini merupakan kontribusi lebih lanjut dari review terhadap kualitas produk yang lebih baik. Berikut ini, metode berbagai review disajikan. Studi perbandingan metode kajian tim adalah subyek Pasal 8.4; pendapat pakar dibahas dalam Bagian 8.5.

8.2. Rancangan Formal Review

Formal desain review, bervariasi disebut "review desain", "DRs" dan "formal Review teknis (FTR) ", berbeda dari semua instrumen review lainnya dengan menjadi hanya meninjau yang diperlukan untuk persetujuan desain produk. Tanpa persetujuan ini, tim pengembangan tidak dapat melanjutkan ke yang berikutnya fase dari proyek pengembangan perangkat lunak. Review desain formal mungkin dilakukan pada setiap selesai tonggak perkembangan yang memerlukan suatu analisis atau desain dokumen, apakah dokumen yang dipersyaratkan spesifikasi atau rencana instalasi. Daftar umum desain formal Review diberikan dalam Bingkai 8.2.

Tabel 8.1. Beberapa rancangan review formal yang umum

Singkatan	Keterangan
DPR	Development Plan Review
SRSR	Software Requirement Specification Review
PDR	Preliminary Design Review
DDR	Detailed Design Review
DBDR	Data Base Design Review
TPR	Test Plan Review
STPR	Software Test Procedure Review
VDR	Version Description Review

OMR	Operator Manual Review
SMR	Support Manual Review
TRR	Test Readliness Review
PRR	Product Release Review
IPR	Installation Plan Review

Pembahasan yang dilakukan dalam rancangan formal review akan fokus terhadap :

- Partisipan
- Persiapan sebelumnya
- Tahap merancang review
- Rekomendasi kegiatan setelah review

8.2.1. Partisipan dalam rancangan review

Review dilaksanakan oleh seorang pimpinan review dan anggota tim yang harus mempunyai persyaratan sebagai berikut :

- Pimpinan
  - Mempunyai pengetahuan dan pengalaman dalam pengembangan sebuah proyek dan tipe review. Persiapan awal untuk mempelajari proyek yang sedang berlangsung tidak perlu dilakukan.
  - Di level sederajat atau lebih tinggi dari project leader
  - Mempunyai hubungan yang baik dengan project leader dan timnya
  - Posisinya merupakan sisi eksternal dari tim proyek
- Anggota tim
  - Keseluruhan anggota tim harusnya dipilih dari para anggota senior dari tim proyek bersama-sama dengan profesional senior lain yang ditetapkan dalam proyek yang lain.

8.2.2. Persiapan rancangan review

Tugas utama dari pimpinan tim pada tahap persiapan review adalah :

- Melakukan kesepatakan dengan semua anggota tim
- Menjadwalkan tahap review
- Mendistribusikan rancangan dokumen diantara anggota tim (hard copy, file elektronik dll)

Ini sangat penting bahwa sidang peninjauan dijadwalkan segera setelah dokumen desain telah didistribusikan kepada anggota tim review. Sesi tepat waktu panjang tidak masuk akal mencegah waktu

dari elapsing sebelum tim proyek dapat dimulai dalam tahap pengembangan selanjutnya dan dengan demikian mengurangi resiko pergi jadwal.

Persiapan Tim Pengembang

Kewajiban utama tim sebagai pendekatan Review sesi adalah untuk menyiapkan pendek penyajian dokumen desain. Dengan asumsi bahwa tim review anggota telah membaca dokumen desain menyeluruh dan sekarang akrab dengan proyek garis besar, presentasi harus fokus pada profesional utama masalah menunggu persetujuan daripada membuang-buang waktu deskripsi proyek secara umum.

Tip Implementasi

Salah satu teknik yang paling umum digunakan oleh para pemimpin proyek untuk menghindari kritik profesional dan melemahkan efektivitas review adalah komprehensif penyajian dokumen desain. Jenis presentasi unggul dalam waktu yang mengkonsumsi. Ini knalpot tim review dan daun sedikit waktu, jika ada, untuk diskusi. Semua pemimpin review berpengalaman tahu bagaimana menangani fenomena ini. Dalam kasus di mana pemimpin proyek berfungsi sebagai pemimpin review, seseorang dapat mengamati terutama ampuh taktik ditujukan stymieing Review yang efektif: pengangkatan tim review yang besar dikombinasikan dengan presentasi yang komprehensif dan panjang.

8.2.3. Tahap perancangan review

Pimpinan review harus mempunyai pengalaman dalam memimpin diskusi dan memasukkan ke dalam agenda kunci keberhasilan pada tahap perancangan review termasuk :

- Presentasi pendek tentang rancangan dokumen
- Komentar yang dibuat oleh anggota tim review
- Verifikasi dan validasi tiap komentar kemudian didiskusikan untuk menentukan tindakan yang diperlukan (perbaikan, perubahan dan penambahan) dimana tim proyek harus melakukannya.
- Keputusan tentang rancangan produk (dokumen), yang menentukan kemajuan proyek dan dibagi menjadi tiga bentuk :
    - Disetujui penuh
    - Disetujui sebagian
    - Ditolak

8.2.4. Kegiatan setelah review

Selain dari laporan DR, tim DR atau perwakilannya diperlukan untuk menindaklanjuti kinerja koreksi dan untuk menguji dikoreksi bagian.

Laporan DR

Salah satu tanggung jawab pemimpin review adalah untuk mengeluarkan laporan DR segera setelah sesi review. Awal distribusi laporan DR memungkinkan tim pengembang untuk melakukan koreksi sebelumnya dan meminimalkan penjaga penundaan jadwal proyek. Isi dari laporan yang utama terdiri dari :

- Ringkasan dari diskusi review
- Keputusan tentang keberlanjutan proyek
- Satu daftar penuh berisi tindakan yang diperlukan tentang perbaikan, perubahan dan penambahan yang harus dilakukan oleh tim proyek. Untuk tiap kegiatan tanggal kelengkapan dan anggota tim yang bertanggung jawab harus ditulis
- Nama dari anggota tim yang ditetapkan untuk melakukan perbaikan.

Infrastruktur rancangan review :

- Mengembangkan cek list untuk tiap tipe rancangan dokumen atau paling tidak satu untuk masing-masing.
- Melatih senior profesional untuk merawat proses review dianggap sebagai pekerjaan teknis yang utama. Pelatihan profesional melayani tim perancang review.
- Secara periodik menganalisis keefektifan perancangan review sebelumnya berdasarkan temuan kesalahan untuk meningkatkan metodologi perancangan review.
- Jadwal dari perancangan review merupakan bagian dari perencanaan proyek dan mengalokasikan sumber daya yang diperlukan sebagai satu kesatuan dalam prosedur standar sebuah perusahaan dalam pengembangan software

8.3. Peer Review

Dua metode peer review, inspections dan walkthrough, dibahas dalam bagian ini. Perbedaan utama antara review desain formal dan metode peer review berakar pada peserta dan otoritas. Sementara sebagian besar peserta di DRs memegang posisi lebih tinggi dari pimpinan proyek dan perwakilan pelanggan, peserta dalam peer review sama dengan pimpinan proyek, dan anggota dari departemennya

dan unit lainnya. Perbedaan utama lainnya terletak pada derajat kewenang dan tujuan masing-masing metode review. Tinjauan desain formal ditujukan untuk menyetujui desain dokumen sehingga pekerjaan pada tahap selanjutnya pada suatu proyek dapat dimulai. Kewenangan ini tidak ada di peer review, yang tujuan utamanya adalah mendeteksi kesalahan dan penyimpangan dari standar.

Saat ini,munculnya alat desain yang terkomputerisasi, diantaranya CASE tool dan sistem dari paket perangkat lunak yang luas, membuat beberapa profesional cenderung mengurangi cara review manual seperti inspections dan walkthrough. Namun demikian, survei perangkat lunak di masa lalu serta penelitian empiris baru-baru ini telah memberikan banyak bukti yang meyakinkan bahwa peer review merupakan metode yang sangat efektif dan efisien.

Yang membedakan walkthrough dan inspection adalah tingkat formalitas,dengan pemeriksaan yang lebih formal dari keduanya. Inspection menekankan tujuan dari koreksi. Sedangkan temuan walkthrough adalah terbatas hanya pada pemberian komentar terhadap review sebuah dokumen, temuan inspection juga dimasukkan ke dalam upaya untuk mengembangkan metode per se. inspection, merupakan lawan dari walkthrough, oleh karena itu dianggap lebih berkontribusi untuk level yang lebih umum dari SQA.

Pemeriksaan biasanya didasarkan pada infrasuktur yang meliputi banyak hal diantaranya :

- Pengembangan dari perkembangan inspection checklists untuk tiap-tiap desain dokumen serta bahasa coding dan tool, yang secara berkala diperbaharui.
- Pengembangan dari jenis tabel defect type frequency, berdasarkan temuan terakhir, yang diarahkan pada inspeksi ke potensi "defect concentration areas".
- Pelatihan profesional yang kompeten dalam masalah proses pemeriksaan, sebuah proses yang memungkinkan mereka untuk dilayani sebagai kepala inspeksi (moderator) ataupun anggota tim inspeksi. Para karyawan yang terlatih melayani sebagai seorang reservoir dari profesional inspektor untuk proyek-proyek yang akan datang.
- Analisis berkala terhadap efektivitas pemeriksaan yang lalu guna peningkatan metode pemerikasaan.
- Memasukkan pemeriksaan terjadwal ke dalam rencana kegiatan proyek dan alokasi sumber daya yang diperlukan, termasuk sumber daya untuk perbaikan cacat.

Proses inspection dan proses walkthrough yang dijelaskan di sini adalah versi umum. Organisasi seringkali memodifikasi metode ini, dengan adaptasi yang mencerminkan "local color", yaitu karakter

dari suatu pengembangan dan unit-unit SQA, produk software yang dikembangkan, struktur dan komposisi tim, dsb. Penting bahwa dalam tanggapan terhadap variasi ini, terutama dalam prosedur walkthrough, perbedaan antara kedua metode tersebut semakin samar. Keadaan ini telah meyakinkan beberapa ahli untuk melihat walkthrough sebagai sebuah tipe inspection begitu juga sebaliknya.

Perdebatan tentang metode mana yang lebih baik terus berlangsung, dengan dimana masing-masing pihak berpendapat berdasarkan keunggulan dari pendekatan-pendekatan yang mereka sukai. Berdasarkan survei pada masing-masing metode, Gilb dan Graham (1993) menyimpulkan bahwa sebagai alternatif untuk inspection, walkthrough ditampilkan “far fewer defects found but at the same cost”.

Pembahasan tentang peer review nantinya akan diutamakan pada :

- Peserta peer review.
- Persyaratan guna persiapan peer review.
- Sesi peer review.
- Kegiatan Pasca-peer review.
- Efisiensi peer-review
- Cakupan dari peer-review

Dengan sedikit adaptasi, prinsip-prinsip dan proses peer desain review juga dapat berhasil diterapkan pada kode peer review. Desain dan kode inspection, sebagai model prosedural, pada awalnya digambarkan dan dirumuskan oleh Fagan (1976, 1986). Sedangkan pada walkthrough, Yourdon (1979) menyediakan pembahasan yang menyeluruh dan rinci tentang prinsip-prinsip dan proses yang terkait.

8.3.1. Partisipan dalam peer review

Tim peer review yang baik terdiri dari 3-5 peserta. Dalam kasus-kasus tertentu, penambahan satu hingga tiga peserta bisa dilakukan. Semua peserta harus mengenal pembuat sistem perangkat lunak. Faktor utama yang berperan terhadap keberhasilan peer review adalah kelompok "blend" (yang membedakan antara inspecton dan walkthrough).

Tim peer review yang dimaksud termasuk :

- Pimpinan review
- The author
- Specialized professionals.

Pimpinan review

Peran pimpinan review ("moderator" dalam inspection, "coordinator"di walkthrough) hanya berbeda sedikit dengan jenis peer review. Calon pada posisi ini harus :

1. Mahir dalam pengembangan proyek dengan tipe-tipe tertentu dan akrab dengan teknologi tersebut. Pengenalan dengan proyek .
2. Memiliki hubungan yang baik dengan author dan tim pengembang.
3. Orang diluar tim proyek.
4. Terbukti berpengalaman dalam hal koordinasi dan kepemimpinan dalam suatu rapat profesional.
5. Untuk inspection, pelatihan sebagai moderator juga diperlukan.

Author

author selalu menjadi peserta pada setiap jenis peer review.

Specialized professionals

Para specialized professional ikut dalam dua metode peer review berbeda. Ciri-ciri profesional untuk inspection adalah :

- Desainer: sistem analis bertanggung jawab pada analisis dan desain sistem perangkat lunak.
- Seorang coder atau pelaksana: seorang profesional bertugas pada bagian coding, sebaiknya ditunjuk menajdi pimpinan tim design coding. disini inspector diharapkan dapat berperan dalam mendeteksi cacat yang dapat menyebabkan error dan kesulitan implementasi software.
- Seorang tester: seorang tester profesional sebaiknya ditunjuk sebagai pemimpin tim penguji, yang fokus pada identifikasi kesalahan desain yang biasanya terdeteksi selama tahap pengujian.

Ciri-ciri professional untuk walkthrough adalah :

- pelaksana standar. anggota dari tim ini adalah mereka yang ahli dalam pengembangan standar dan prosedur, mereka diberikan tugas mencari penyimpangan dari standar-standar dan prosedur. Kesalahan jenis ini secara substansial mempengaruhi efektivitas tim secara jangka panjang, pertama karena mereka menyebabkan kesulitan tambahan bagi anggota baru untuk bergabung dengan tim pengembangan, dan kemudian karena mereka akan mengurangi efektivitas tim yang akan memelihara sistem.
- Seorang ahli pemeliharaan yang diikutsertaan untuk fokus pada pemeliharaan, fleksibilitas dan testabi bug atau perubahan-perubahan yang mungkin terjadi. Bidang lain yang membutuhkan keahliannya adalah dokumentasi, dimana kelengkapan dan kebenaran sangat penting untuk kegiatan pemeliharaan.

- Seorang user. Partisipasi internal (ketika pelanggan merupakan kesatuan unit dalam sebuah perusahaan) atau eksternal user di tim walkthrough berkontribusi dalam hal validasi karena mereka bertugas menguji sistem perangkat lunak dari sudut pandang userconsumer bukan dari sudut pandang designer–supplier. Dalam kasus di mana seorang user tidak ada, seperti dalam pengembangan perangkat lunak COTS, sebuah anggota tim dapat mengambil peran dan fokus pada masalah validasi dengan membandingkan persyaratan asli dengan desain yang sebenarnya.

Tim penguji

Melakukan sesi review membutuhkan, kealamian, pengalihan tugas tertentu kepada anggota tim. Dua dari anggota tersebut adalah presenter dokumen dan juru tulis, yang mendokumentasikan diskusi.

- Presenter. Selama sesi inspection, presenter guna dokumentasi dipilih oleh moderator, biasanya, presenter bukanlah penulis dokumen. Dalam banyak kasus seorang software coder berperan sebagai presenter karena dia adalah anggota tim yang paling memahami logika desain dan implikasinya untuk coding. Sebaliknya, pada sebagian besar sesi walkthrough, yang berperan adalah author, pihak yang paling mengerti tentang dokumen, yang dipilih untuk bergabung ke dalam grup. Beberapa ahli mengtakan bahwa author yang ditugaskan sebagai presenter bisa mempengaruhi keseimbangan anggota kelompok, sehingga mereka berpendapat bahwa presenter dari pihak "Netral" adalah yang lebih baik.
- Juru tulis. Pemimpin tim akan sering - tetapi tidak selalu - berperan sebagai juru tulis untuk suatu sesi, dan mencatat bagian-bagian yang akan diperbaiki oleh tim pengembangan. Tugas ini lebih dari sebuah prosedural, melainkan membutuhkan pemahaman yang menyeluruh pada topik yang dibahas.

8.3.2. Persiapan untuk tahap peer review

Pimpinan review dan anggota tim harus matang dalam hal persiapan, dimana jenis review menentukan ruang lingkup mereka.

Persiapan pimpinan peer review untuk sesi review

Tugas utama pimpinan review pada tahap persiapan adalah :

- Bekerjasama dengan author menentukan bagian mana dari dokumen desain yang harus diriview. Bagian tersebut bisa berupa:
  - ✓ Bagian yang paling sulit dan kompleks
  - ✓ Bagian yang paling penting, di mana setiap cacat dapat menyebabkan kerusakan parah untuk aplikasi program dan begitu juga untuk usernya

- ✓ Bagian yang rentan terhadap cacat.

- Menjadwalkan sesi peer review. Dianjurkan untuk membatasi sesi review hanya dua jam, oleh karena itu beberapa sesi review terjadwal (2 sesi dalam sehari) ketika tugas review dapat dipadatkan. Penting untuk menjadwalkan sesi setelah bagian desain dokumen yang dimaksud siap untuk pemeriksaan. Perilaku ini cenderung untuk meminimalkan ruang lingkup dan/atau jumlah penambahan desain berdasarkan bagian dokumen yang nantinya ditemukan cacat pada penjadwalan review. Selain itu, untuk proses yang berjalan lancar, pemimpin review harus menjadwalkan sebuah rapat ringkasan untuk timnya.
- Untuk mendistribusikan dokumen kepada anggota tim sebelum sesi review.

Peer review tim persiapan untuk sesi review

Persiapan yang diperlukan oleh anggota tim inspection cukup lengkap,sedangkan yang diperlukan oleh anggota tim walkthrough cukup sederhana.

Anggota tim inspection diharapkan membaca bagian dokumen yang akan direview dan mendaftar komentar-komentar sebelum sesi inspection dimulai. Persiapan awal ini dimaksudkan untuk menjamin efektivitas sesi tersebut. Mereka juga akan diminta untuk berpartisipasi dalam rapat ringkasan. Pada pertemuan ini, author menyiapkan anggota tim inspection yang memiliki latar belakang yang sesuai untuk meriview bagian dokumen yang dipilih: proyek secara umum, logika, proses, keluaran, masukan, dan antarmuka. Dalam kasus dimana para peserta sudah mengenal baik dengan materi, sebuah rapat ringkasan bisa ditutup.

Sebuah alat penting yang mendukung inspection review adalah checklist. Dalam departemen pengembangan yang bagus, orang yang dapat menemukan daftar khusus dipilih untuk pengembangan dokumen dengan jenis yang lebih umum (lihat Bab 15).

Sebelum sesi walkthrough, anggota tim membaca materi bertujuan untuk mendapatkan gambaran umum dari bagian yang akan direview, dalam hal proyek dan lingkungannya. Peserta yang kurang tahu tentang proyek tersebut dan substantifnya akan butuh waktu persiapan yang lebih. Dalam kebanyakan organisasi yang menerapkan walkthrough, tim peserta tidak perlu untuk menyiapkan pandangan awal mereka.

8.3.3. Tahap peer review

Ciri dari peer review seperti pada formulir berikut. Presenter membaca bagian dari dokumen dan menambahkan, jika diperlukan, sebuah penjelasan singkat tentang topik yang dibahas. Pada ssat sesi berlangsung, peserta sebaiknya memberikan pendapat mereka tentang dokumen tersebut atau mengarahkan reaksi mereka dalam bentuk komentar-komentar. Diskusi harus dibatasi hanya untu mengidentifikasi kesalahan, yang berarti bahwa bagian ini bukanlah proses pencarian solusi. Tidak seperti sesi inspection, agenda sesi walkthrough dibuka dengan presentasi singkat author atau gambaran dari proyek dan bagian desain yang akan direview.

Selama sesi, juru tulis harus mendokumentasikan setiap kesalahan berdasarkan lokasi dan deskripsi, tipe dan karakter (kesalahan, bagian yang hilang atau bagian yang berlebih). Juru tulis sesi inspecsion akan menuliskan tingkat kesalahan masing-masing cacat, faktor yang akan digunakan dalam analisis statistik cacat yang ditemukan dan untuk perumusan tindakan preventif dan korektif. Klasifikasi tingkat kesalahan dituliskan dalam Lampiran C MIL-STD-498 (DOD, 1994) dan disajikan di Tabel 8.1, yang memaparkan kerangka kerja untuk mengklasifikasi tingkat kesalahan.

Mengenai lama sesi inspeksi dan walkthrough, peraturan yang sama berlaku untuk DRs: sesi tidak boleh melebihi dua jam, atau dijadwalkan lebih dari dua kali sehari. Pressman's "golden guidelines" for conducting successful DR sessions akan berguna (lihat Frame 8.3).

Sesi dokumentasi

Dokumentasi yang dihasilkan pada akhir sesi inspection jauh lebih komprehensif daripada sesi walkthrough.

Berikut adalah Dua dokumen yang dihasilkan setelah sesi inspekction dan
kemudian dibagikan kepada peserta :

1. Laporan inspection sessuin findings. Laporan ini, dibuat oleh juru tulis, yang harus diselesaikan dan diberikan tepat setelah sesi selesai. Tujuan utamanya adalah untuk memastikan dokumentasi penuh pada kesalahan yang ditemukan yang nantinya akan dikoreksi dan ditindaklanjuti. Salah satu contoh laporan terdapat di Lampiran 8B.

Tabel 8.1 Klasifikasi desain error berdasarkan tingkat keparahan

Severity	Description
5 (critical)	(1) Prevents accomplishment of essential capabilities. (2) Jeopardizes safety, security or other critical requirements.
4	(1) Adversely affects the accomplishment of essential capabilities, where no work-around solution is known. (2) Adversely affects technical, cost or schedule risks to project or system maintenance, where no work-around solution is known.
3	(1) Adversely affects the accomplishment of essential capabilities, where a work-around solution is known. (2) Adversely affects technical, cost or schedule risks to the development project or to the system maintenance, where a work-around solution is known.
2	(1) User/operator inconvenience that does not affect required mission or operational essential capabilities. (2) Inconvenience for development or maintenance personnel, but does not prevent the realization of those responsibilities.
1 (minor)	Any other effect.

Source: After DOD (1994)

2. Laporan inspection session summary. Laporan ini harus disusun oleh pemimpin inspection tepat setelah sesi atau serangkaian sesi berhubungan dengan dokumen yang sama. Sebuah laporan khusus dari jenis ini merangkum inspection findings dan sumber daya yang digunakan dalam inspection, dan menyajikan kualitas dasar dan metrik efisiensi. Laporan ini melayani utamanya sebagai masukan untuk tujuan analisis pada peningkatan proses pemeriksaan dan tindakan korektif yang melampaui dokumen atau proyek tertentu. Contoh Laporan inspection session summary terdapat di Lampiran 8C.

Pada akhir sesi atau serangkaian sesi walkthrough, salinan dari dokumentasi kesalahan - yang "laporan walkthrough session findings" – harus diserahkan kepada tim pengembangan dan peserta sesi.

8.3.4. Kegiatan setelah peer review

Unsur mendasar yang membedakan antara kedua metode peer review dibahas di sini adalah topik post-peer review.

Proses pemeriksaan, berlawanan dengan walkthrough, tidak berakhir dengan sesi review ataupun distribusi laporan. kegiatan Pasca-inspeksi harus dilakukan untuk membuktikan :

- Koreksi cepat dan efektif dan perbaikan semua kesalahan oleh desainer/author dan timnya, seperti yang dilakukan oleh pemimpin inspection (atau anggota tim yang lain) dalam bentuk tindak lanjut suatu kegiatan.
- Transmisi pemeriksaan laporan ke Tindakan Korektif internal Board untuk analisis. Tindakan ini merupakan awal perbaikan dan pencegahan tindakan yang akan mengurangi kesalahan kedepannya dan meningkatkan produktivitas.

8.3.5. Efisiensi dari peer review

Masalah efisiensi pendeteksian cacat pada metode peer review yang tepat dan perbandingan dengan metode deteksi cacat SQA yang lebih terus menerus diperdebatkan. Beberapa metrik yang lebih umum diterapkan untuk estimasi efisiensi dari peer review, seperti yang disarankan dalam literatur, adalah:

- Peer review deteksi efisiensi (rata-rata jam kerja per cacat terdeteksi).
- Peer review deteksi cacat kepadatan (jumlah rata-rata cacat yang ditemukan per halaman dari dokumen desain).
- Internal peer review efektivitas (persentase cacat yang dideteksi oleh peer review sebagai persentase dari total cacat yang terdeteksi oleh pengembang).

8.3.6. Cakupan peer review

Hanya sebagian kecil dokumen dan kode yang pernah mengalami peer review. Cakupannya sekitar 5-15% dari halaman dokumen masih merupakan bagian yang signifikan terhadap jumlah kualitas desain karena factor yang menentukan manfaat peer review terhadap kualitas total bukanlah persentase halaman yang terjamah tetapi halaman yang dipilih. Yang terpenting dengan meningkatnya penggunaan perangkat lunak daur ulang, jumlah halaman dokumen dan baris kode yang butuh diinspection semlibatkan dalam peer review sehingga kesalahan dapat dihilangkan.

8.4. Perbandingan Metode-Metode Tim Review

Untuk praktisi dan analis diberikan, perbandingan dari ketiga metode tim review yang dibahas dalam bab ini. Tabel 8.4 menyajikan perbandingannya.

Tabel 8.3: Kefektifitasan kode pemeriksaan di Fujitsu menurut Cusumano

Tahun	Metode Penditeksi Cacat			Cacat per 1000 baris dari kode yang dipertahankan
	Test %	Disain review %	Code inspection %	
1977	85	-	15	0.19
1978	80	5	15	0.13
1979	70	10	20	0.06
1980	60	15	25	0.05
1981	40	30	30	0.04
1982	30	40	30	0.02

Frame 8.4 Bagian yang direkomendasikan untuk dimasukkan atau dihilangkan dari peer reviews

Bagian yang direkomendasikan untuk dimasukan:	Bagian yang direkomendasikan untuk dihilangkan :
• Bagian dari logika rumit • Bagian kritis • Bagian yang berurusan dengan lingkungan baru • Bagian yang dirancang oleh anggota tim baru atau anggota tim yang belum berpengalaman	• Bagian langsung(Tanpa komplikasi) • Bagian dari sebuah tipe yang sudah ditinjau beberapa kali oleh tim dalam proyek-proyek yang sama di masa lalu. • Bagian yang, rusak, tidak diharapkan mempengaruhi fungsionalitas • Pemakaian kembali disain dan kode • Pengulangan bagian dari disain dan kode

Properties	Formal design reviews	Inspections	Walkthroughs
Tujuan utama langsung	• Mendeteksi error • Identifikasi resiko baru • Menyetujui disain dokumen	• Mendeteksi error • Mengidentifikasi penyimpangan dari standardnya	• Mendeteksi error
Tujuan utama tidak langsung	• Pertukaran pengetahuan	• Pertukaran pengetahuan • Mendukung tindakan korektif	• Pertukaran pengetahuan
Pemimpin review	• Kepala software engineering atau anggota staff senior	• Moderator terlatih (peer)	• koordinator(peer, pemimpin proyek pada suatu kesempatan)
Peserta	• Staff level atas • Wakil customer	• Kawan sesama	• Kawan sesama

Partisipasi pemimpin proyek	Ya	Ya	Ya, selalu sebagai inisiator review
Keahlian khusus pada team	-	Designer Pemrogram atau pelaksana Penguji	Standar yang dilaksanakan Ahli pemeliharaan Perwakilan pengguna

Proses dari review

Properties	Formal design reviews	Inspections	Walkthroughs
Gambaran rapat	Tidak	Ya	Ya
Persiapan peserta	Ya, menyeluruh	Ya, menyeluruh	Ya,singkat
Sesi review	Ya	Ya	Ya
Mengijinkan koreksi	Ya	Ya	Ya

Infrastruktur

Properties	Formal design reviews	Inspections	Walkthroughs
Training formal dari peserta	Tidak	Ya	Tidak
Menggunakan daftar	Tidak	Ya	Tidak
Kesalahan yang terkait dengan pengumpulan data	secara formal tidak diperlukan	Secara formal diperlukan	secara formal tidak diperlukan
Dokumentasi review	Laporan formal design review	• inspection session findings report. • Inspection session summary report	Walktrough session findings report

8.5. Pendapat Para Ahli

Metode review terakhir yang kita bahas adalah penggunaan dari pendapat ahli. Pendapat ahli, disiapkan oleh para ahli dari luar, mendukung kualitas evaluasi dengan memperkenalkan kemampuan tambahan kepada staff review internal, mutu dari aktivitas organisasi internal semakin diperkuat. Ahli luar mengajarkan keahliannya dengan cara :

- Mempersiapkan pendapatnya tentang sebuah dokumen atau bagian kode.

- Berpartisipasi sebagai seorang anggota dari sebuah internal design review, inspection atau walkthrough team.

Pendapat seorang ahli luar serta partisipasinya sebagai anggota eksternal dari review team paling menguntungkan disituasi berikut :

- Kurangnya kemampuan ahli yang ada di area khusus
- Dengan kurangnya kemampuan ahli akan menyebabkan keterlambatan pada penyelesaian jadwal proyek akibat tekanan beban kerja.
- Keraguan yang disebabkan oleh besarnnya ketidaksetujuan diantara ahli senior organisasi.
- Dalam organisasi kecil, dimana jumlah calon yang sesuai untuk team review tidak mencukupi.

8.6. Ringkasan

1) Jelaskan tujuan langsung dan tidak langsung dari metodologi review !

2) Jelaskan kontribusi dari para ahli dari luar terhadap performa tugas review !

3) Bandingkan tujuan dan partisipan dari metode review tiga tim !

8.7. Apendik 8A : Rancangan Review untuk Form Laporan

LAPORAN DESIGN REVIEW

tanggal DR _____laporan disajikan untuk_____
Nama project: _____
Dokumen review: _____ Versi: _____
Team review: _____

1. ringkasan yang didiskusikan

#	Judul diskusi	Jumlah jenis tindakan

2 Jenis tindakan

#	Jenis tindakan yang dilakukan	Tanggung jawab pekerja	Tanggal penyelesaian	Persetujuan penyelesaian	
				Tanggal	Tanda tangan

3 Keputusan tentang design produk

- Persetujuan penuh
- Persetujuan Sebagian. Persetujuan diberikan untuk kelanjutan ke fase berikutnya dari bagian berikut:
- Persetujuan penolakan

Komentar:

Laporan disetujui oleh :

Nama dari peserta	Tanggal	Tanda tangan	Nama dari peserta	tannggal	Tanda tangan

8.8. Apendik 8B : Tahap Pemeriksaan Temuan Form Laporan

INSPECTION SESSION FINDING REPORT

Tanggal sesi: _____ Laporan disiapkan untuk: _____
Nama projek: _____
Dokumen diperiksa: _____ Versi : _____
Sesi dokumen diperiksa: _____
Inspection team: _____

1 Daftar error

#	Type error	Error alami (W/M/E)*	Diskripsi error	Lokasi error	Error yang parah

2 Tindak lanjut dari pilihan	
a	Tindak lanjut yang dilakukan oleh:
b	Re-inspeksi dianjurkan: Ya/Tidak
c	

3 Komentar

*W = Wrong M = Missing E = Extra

8.9. Apendik 8C : Tahap Pemeriksaan Kesimpulan Laporan

Goldenbug Ltd.

INSPECTION SESSION SUMMARY REPORT

Tanggal sesi: _____

Nama projek: _____

Dokumen yang diinspeksi: _____ Versi: _____

Sesi dokumen yang diinspeksi: _____ Total: (A) _____ halaman/k baris tesk

Tim Inspection :_____

1 Sumber yang diinvestasikan(jam kerja)

#	Anggota team	Tinjauan pertemuan	persiapan	Sesi inspeksi	Total(jam)	Komentar
1	Pemimpin inspeksi Anita	1	3	2.5	6.5	Memasukan persiapan laporan
2	John	1	4	2	7	
3	Ben	1	4	2	7	
4						
5						
	Total	3	11	6.5	(B)20.5	

2 Ringkasan error

Error terberat	Error alami			Total error	Faktor terberat	Total error(standar)	Komentar
	W	M	E*				
5-kritis	1			1	16	16	
4			2	2	8	16	
3	3			3	4	12	
2		2		2	2	4	
1-kecil	4	1	2	7	1	7	
Total	8	3	4	(c)15		(D)53	

3 metrik deteksi cacat

(1) Rata-rata cacat perhalaman = C/A =15/31=0.48

(2) Rata-rata cacat perhalaman (standar) = D/A =53/31= 1.71

(3) Efesiensi deteksi cacat (cacat per jam) = B/C =20.5/15= 1.37

(4) Efesiensi Standar deteksi cacat (jam per sstandar cacat) =B/D =20.5/53= 0.39

Disiapkan oleh: _____ Tanda tangan: _____ Tanggal: _____

*W = Wrong M = Missing E = Extra

Bab9. Strategi Testing Software

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan tujuan testing
2. Mengetahui perbedaan antara strategi testing, keuntungan dan kerugiannya
3. Menggambarkan konsep dari *black box* testing dan *white box* testing dengan baik
4. Mendefinisikan *path coverage* dan *line coverage*
5. Mendeskripsikan macam-macam tipe *black box* testing

9.1. Definisi dan Tujuan

Berbagai macam definisi dari software testing dapat dijumpai pada banyak literature dan berbagai macam lingkup proses. Berdasarkan Myers(1997, bab 10) :

"Testing adalah proses mengeksekusi sebuah program yang bertujuan untuk menemukan eror"

Berdasarkan definisi ini aktifitas pengujian dilakukan oleh seorang pemimpin dan rekan kerjanya untuk menjalankan software, seperti halnya test yang dijalankan oleh tim penguji. Definisi yang lebih formal adalah dua definisi testing yang dikemukakan oleh IEEE Std 610.12 (IEEE, 1990) :

1) Proses pengoperasian sebuah sistem atau komponen terhadap kondisi-kondisi yang telah ditetapkan, mengamati dan memperhatikan hasil, dan membuat evaluasi dari beberapa aspek dari sistem atau komponen tersebut.
2) Proses menganalisa sebuah software untuk mendeteksi perbedaan antara software tersebut dengan kemauan user dan untuk mengevaluasi fitur-fitur dari software tersebut.

Yang patut diingat bahwa menurut definisi yang kedua, menjalankan program sebagai bagian dari testing tidak diperkenankan. Definisi yang terdapat pada buku ini menekankan pada karakteristik pengoperasian resting. Lihat frame 9.1.

Frame 9.1 Software test – Pengertian

software testing adalah suatu proses **formal** yang dilakukan oleh **tim khusus** (tim spesialis penguji) yang mana satu unit software, beberapa gabungan dari unit-unit software atau keseluruhan paket dari software akan diuji dengan cara **menjalankan program di komputer**. Keseluruhan rangkaian dari pengujian dilakukan berdasarkan **prosedur yang telah disetujui** sebelumnya dalam **test case yang telah disepakati**

TESTING :

- Proses menjalankan program dengan maksud mencari kesalahan (definisi klasik)
- Proses dari sistem operasi atau komponen dibawah kondisi yang ditetapkan, mengamati atau mencatat hasil dan membuat evaluasi terhadap beberapa aspek dari sistem atau komponen. (IEEE, 1990)
- Proses menganalisis sebuah item software untuk mendeteksi perbedaan antara hasil yang ada dan kondisi yang dibutuhkan. (IEEE, 1990)

SOFTWARE TESTING

Proses formal yang dihasilkan oleh tim spesialis testing unit software, beberapa unit software terintegrasi atau keseluruhan paket software yang diperiksa dengan cara menjalankan program pada sebuah komputer. Semua jenis testing dilakukan berdasarkan prosedur testing yang disepakati.

Berdasarkan kata-kata yang ditekankan pada definisi diatas kita dapat membandingkan kaakteristik dari software testing dengan jenis tool jaminan kualitas mutu software yang lain :

- Formal : perencanaan software test merupakan bagian dari project development, dijadwalkan dengan baik dan sering kali dijadikan sebagai poin penting dalam penandatanganan perjanjian kontrak antara user dan developer.
- Tim khusus : tim independen atau konsultan eksternal (diluar developer) yang memiliki keahlian khusus di bidang testing dan dapat melakukan testing secara professional. Biasanya jika test dilakukan oleh developer hasil yang didapat tidak bisa optimal, karena biasanya developer sulit untuk memperbaiki eror yang tidak mereka pertimbangkan sebelumnya. Namun kebanyakan software masih ditest oleh developer yang membuat software itu sendiri.
- Menjalankan program : terdapat beberapa aktifitas dari jaminan mutu software yang tidak menjalankan program, contohnya code inspection, tidak dapat disebut sebagai proses test.
- Prosedur yang telah disetujui : proses testing dilakukan berdasarkan perencanaan dan prosedur yang telah disetujui sebagai SQA prosedur.
- Test case yang telah disepakati : didefinisikan berdasarkan perencanaan. Tidak ada tambahan atu pengurangan dari proses testing. Dengan kata lain jika proses telah dimulai, penguji tidak

diperkenankan untuk mengurangi(menghilangkan) salah satu kondisi testing dan juga tidak diperkenankan untuk menambahi kondisi test baru meskipun itu memungkinkan.

Tujuan Langsung Testing

- Untuk mengidentifikasi dan mengungkapkan sebanyak mungkin kesalahan dalam testing software.
- Untuk membawa software yang ditest, setelah dikoreksi atas kesalahan yang ditemukan, ke level persetujuan terhadap kualitas yang dihasilkan.
- Untuk melakukan testing yang diperlukan secara efektif dan efisien, dalam batas biaya dan waktu.

Tujuan tidak langsung

untuk menyusun daftar eror dari sebuah software.

Yang harus diingat adalah "untuk membuktikan bahwa suatu perangkat lunak telah selesai(siap)" bukanlah sebuah kebetulan. Myers(1979) menyatakan : "Jika tujuan anda adalah untuk menunjukkan tidak adanya eror, maka anda tidak akan menemukan eror tesebut, namun jika tujuan anda adalah untuk menunjukkan jumlah eror yang ada maka anda akan menemukan begitu banyak eror."
Untuk menemukan bug-free software masih sangat mustahil. Oleh karena itu lebih tepatnya kita menyebutnya sebagai "kualitas yang dapat diterima", yang berarti ada beberapa bug/eror yang dapat ditoleransi oleh user. Presentasi tersebut bervariasi berdasarkan software atau user, namun harus lebih kecil dari pada resiko kegagalan tertinggi dari sebuah software.

9.2. Strategi Testing Software

Meskipun metodologi testing sangat beragam, tapi metodologi ini bisa diterapkan dalam sebuah framework tentang dua strategis dasar testing yaitu :

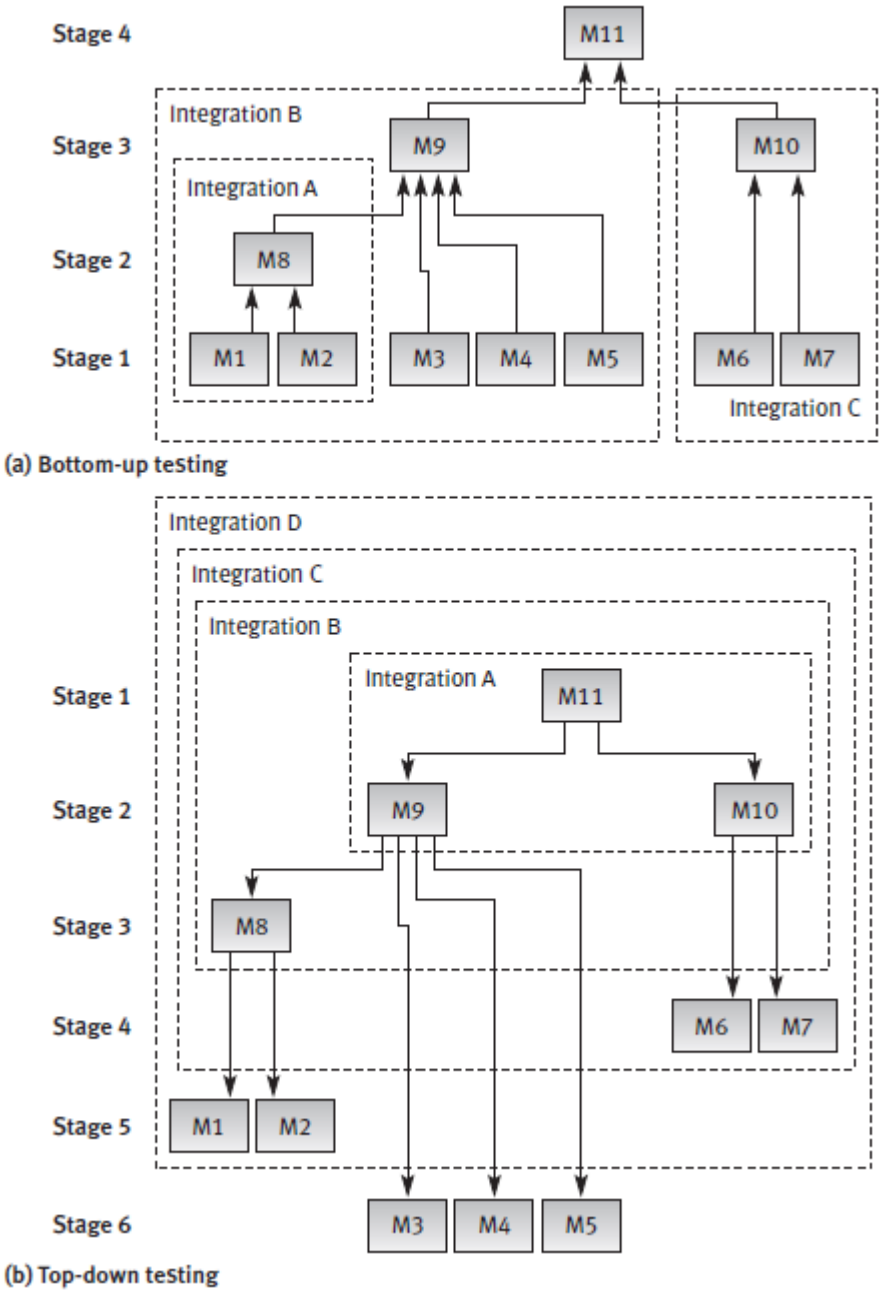
- Melakukan testing secara keseluruhan, sekali paket lengkap maka software siap, jika tidak dikenal sebagai "big bang testing".
- Melakukan testing pada bagian software seperti modul atau unit.

Incremental testing juga dilakukan berdasarkan 2 basic strategi : bottom-up dan top-down. Keduanya berasumsi bahwa sebuat software dibangun secara hierarkikal modul. Pada top-down testing, modul

pertama yang ditest merupakan modul utama dan merupakan level tertinggi dari struktur software, sedangkan modul yang diuji terakhir kali adalah level yang terrendah. Pada bottom-up testing, yang pertama kali diuji adalah level yang terrendah, sedangkan bagian utamanya diuji paling akhir.

Gambar 9.1 menggambarkan tenteng top-down dan bottom-up testing pada pengembangan software dengan 11 modul. Pada gambar 9.1(a) merupakan testing menggunakan bottom-up, dimana prosesnya sebagai berikut :

- Tahap 1 : Unit test modul 1-7
- Tahap 2 : Menguji kesatuan dari modul 1 dan 2 yang terintegrasi dalam modul 8
- Tahap 3: Menguji modul 3,4,5 dan 8 yang terintegrasi dalam modul 9, dan menguji modul 6 dan 7 yang terintegrasi dalam modul 10
- Tahap 4 : Menguji kesatuan sistem modul 9 dan 10 yang terintegrasi dalam modul 11



Gambar 9.1 bottom-up (a) dan top-down(b) testing – ilustrasi(gambaran)

Pada gambar 9.1(b), merupakan pengujian software menggunakan top-down dalam 6 tingkatan. Tampak nyata bahwa perubahan strategi testing menunjukkan adanya perubahan pada penjadwalan test pula. Proses top-down testing, sebagai berikut :

- Tahap 1 : Unit test modul 11
- Tahap 2 : Tes penyatuan A, modul 9 dan 10 yang terintegrasi dalam modul 11

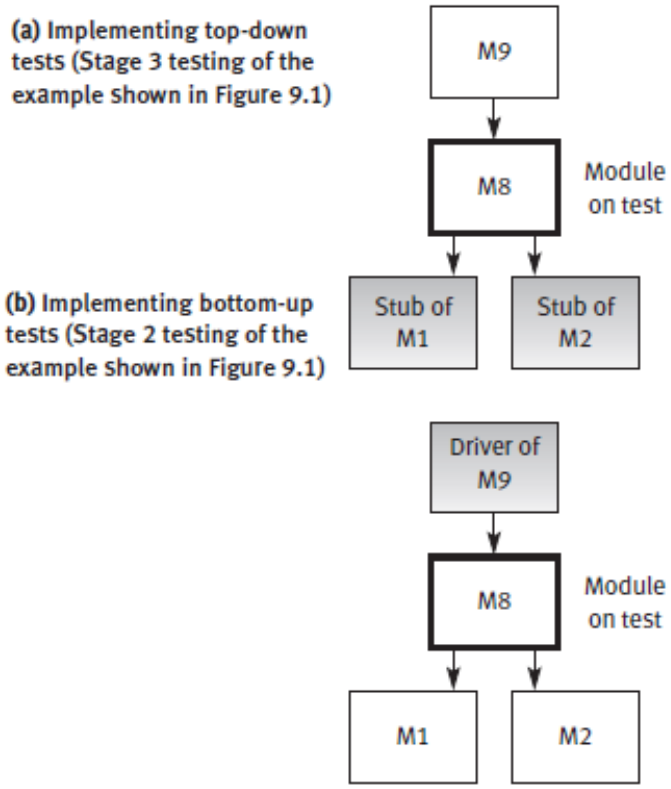
- Tahap 3 : Tes penyatuan B, A disatukan dengan modul 8
- Tahap 4 : Tes penyatuan C, B disatukan dengan modul 6 dan 7
- Tahap 5 : Tes penyatuan D, C disatukan dengan modul 1 dan 2
- Tahap 6 : Sistem D diintegrasikan dengan modul 3,4 dan 5.

Incremental testing yang ditunjukkan pada gambar 9.1 merupakan dua bagian. Bagian dari contoh tersebut adalah “horizontally sequenced” (“breadth first”), dan “vertically squenced” (“depth first”). Jika kita mengubah horizontal path dari top-down sequence yang ditunjukkan pada gambar 9.1(b) ke vertical sequence, testing akan menjadi seperti :

- Tahap 1 : Unit tes modul 11
- Tahap 2 : Integrasi A, modul 11 dan 9
- Tahap 3 : Integrasi B, modul A dengan modul 8
- Tahap 4 : Integrasi C, modul B dengan modul 1 dan 2
- Tahap 5 : Integrasi D, modul C dengan modul 10
- Tahap 6 : Integrasi E, modul D dengan modul 6 dan 7
- Tahap 7 : Sistem tes setelah test E mengintegrasikan dengan modul 3,4 dan 5.

Stubs dan drivers untuk incremental testing

Stubs dan driver adalah simulator pengganti software yang diperlukan jika modul tidak tersedia saat melakukan test. Stub (sering disebut dengan “dummy module”) menggantikan modul terrendah yang tudak tersedia. Stub dibutuhkan untuk top-down testing. Dalam kasus ini, stub menyediakan hasil kalkulasi(perhitungan) modul yang berlevel rendah. Sebagai contoh, pada stage 3 pada gambar top-down testing (gambar 9.1 b), modul 9, yang mengaktifkan modul 8, tersedia; dan telah diuji dan diperbaiki pada stage 2 dalam testing. Stub diperlukan untuk menggantikan level modul yang lebih rendah, yaitu modul 1 dan 2, yang belum lengkap. Testing tersebut terlihat pada gambar 9.1.



Gambar 9.2 penggunaan stub dan driver pada incremental testing – contoh

Seperti stub, driver menggantikan modul namun modul yang levelnya lebih tinggi yang mengaktifkan modul yang akan diuji. Driver melewati data test menuju ke modul test dan menerima hasil dari perhitungannya. Driver dibutuhkan pada bottom-up testing hingga modul dengan level tertinggi didevelop. Contohnya, pada stage 2 dalam bottom-up testing (gambar 9.1 a), level yang lebih rendah, modul 1 dan 2 tersedia, modul-modul tersebut diuji dan diperbaiki pada stage 1 dalam testing. Driver dibutuhkan untuk menggantikan modul dengan level yang lebih tinggi, yaitu modul 9, yang belum lengkap. Keadaan seoerti ini dapat dilihat pada gambar 9.2 (b).

Tips implementasi

Sumber daya yang substansial dapat diperoleh dengan memelihara library stub dan driver untuk kebutuhan yang akan datang.

Bottom-up versus top-down strategi

Keuntungan dari bottom-up strategi adalah meringankan performa dari pengujian tersebut sedangkan kerugiannya adalah keterlambatan dalam mengamati program secara keseluruhan. Keuntungan dari top-down strategi adalah memungkinkan untuk mengetahui keseluruhan fungsi program tak lama setelah pengujian level tertinggi selesai. Dalam beberapa kasus karakteristik ini memungkinkan untuk mengidentifikasi analisis dan design eror, permintaan fungsional, dll. Dan kerugiannya adalah kesulitan dalam membagi menjadi modul-modul untuk pengujian lebih lanjut dan sulit untuk menganalisa hasil pengujian. Para pakar testing masih mendiskusikan tentang strategi mana yang terbaik antara top-down dan bottom-up. Walaupun keputusannya berubah-ubah, namun sejauh ini keputusan masih ada pada tengan developer. Tim penguji harus mengikuti keputusan dari developer karena pengujian merupakan sesuatu yang sangat penting yang dilaksanakan tak lama setelah modul dibuat(dicoding). Pengimplementasian strategi testing yang berbeda dengan strategi pengembangan dapat menyebabkan penundaan jadwal test.

Big bang versus top-down strategi

Kecuali jika program sangat kecil dan sederhana, strategi big bang sangat tidak menguntungkan. Pengidentifikasian eror sangat tidak praktis. Disamping penginvestasian sumber daya yang tinggi, efektivitas dari pendekatan ini relative sangat kecil. Pengidentifikasian eror dengan menggunakan metode ini juga sangat kecil. Selain itu jika dihadapkan dengan suatu paket software, pengenalan error dapat menjadi suatu tugas yang berat karena menyangkut dengan modul yang lain. Batasan seperti inilah yang membuat perkiraan/estimasi dari testing menjadi tidak jelas. Berbeda dengan strategi big bang, incremental testing menawarkan sejumlah keuntungan, yang utama adalah :

- Incremental testing biasanya dilakukan pada modul-modul kecil sebagai unit kesatuan dari test. Karena memudahkan untuk mengidentifikasi presentase eror yang lebih tinggi dibandingkan dengan pengujian software secara keseluruhan
- Lebih sederhana dalam pengidentifikasian dan pengkoreksian eror dan membutuhkan sedikit tenaga kerja karena dilakukan pada volume software yang terbatas

Secara singkat, dalam incremental testing, pengenalan dan pengoreksian eror dilakukan pada tahap lebih awal pada pengembangan dan pengujian, yang mencegah luputnya dalam menemukan eror. Kekurangan dari incremental testing adalah memerlukan banyak programmer untuk mempersiapkan

modul-modul terkecil hingga terbesar dan pengintegrasiannya. Serta kerugian yang lain adalah membutuhkan pengujian yang berulang-ulang untuk program yang sama. (big bang testing hanya memerlukan sekali pengujian).

9.3. Klasifikasi Testing Software

Software Test dapat diklasifikasikan menurut konsep pengujian atau menurut persyaratan klasifikasi berlaku (lihat Bab 3).

9.3.1. Klasifikasi berdasarkan konsep testing

Ada perdebatan mengenai apakah pengujian fungsi perangkat lunak semata-mata menurut output cukup untuk mencapai tingkat kualitas yang dapat diterima. Beberapa menyatakan bahwa struktur internal perangkat lunak dan perhitungan (yaitu, struktur matematika yang mendasarinya, juga dikenal sebagai "mekanisme" perangkat lunak) harus dimasukkan untuk pengujian yang memuaskan. Berdasarkan dua konsep yang berlawanan atau mendekat kepada kualitas software, dua kelas pengujian telah dikembangkan:

1. Pengujian Black Box (fungsionalitas). Mengidentifikasi bug sesuai dengan kegagalan fungsi Software seperti yang terungkap dalam output yang error. Pada kasus output yang ditemukan benar, pengujian kotak hitam mengabaikan perhitungan jalur internal dan pengolahan yang telah dilakukan

2. Pengujian White Box (struktural). Memeriksa perhitungan jalur internal untuk mengidentifikasi bug. Meskipun istilah "putih" dimaksudkan untuk menekankan kontras antara metode dan pengujian black box, metode dengan nama lain - "pengujian Glass Box" - lebih baik dalam mengungkapkan karakteristik dasar, yang menyelidiki kebenaran struktur kode.

Black box testing :

- Melakukan testing dengan mengabaikan mekanisme internal terhadap sebuah sistem atau komponen dan fokus semata-mata terhadap output yang dihasilkan sebagai respon atas input dan kondisi yang dijalankan.
- Testing yang dilakukan untuk mengevaluasi pemenuhan dari sebuah sistem atau kompoenen terhadap kebutuhan fungsional tertentu.

White box testing :

- Testing yang melakukan perhitungan terhadap mekanisme internal dari sebuah sistem atau komponen.

Ketika diterapkan, masing-masing konsep mendekati pengujian software dengan cara yang berbeda, seperi yang akan kita lihat di dalam Sections 9.4 dan 9.5.Dalam banyak kasus kedua konsep tersebut bisa diterapkan, walaupun untuk beberapa kebutuhan SQA hanya satu kelas uji saja yang cocok. Berdasarkan pertimbangan, kebanyakan dari pengujian yang telah dilaksanakan adalah pengujian Black box, yang mana relative sedikit agak mahal.

9.3.2. Klasifikasi berdasarkan persyaratan

Pada bab 3 menerangkan model klasik dari McCall's untuk penggolongan kebutuhan software yang berkualitas. Modelnya telah diperluas kepada penggolongan pengujian yang telah dilakukan untuk memastikan pemenuhan dari kebutuhan masing-masing. Persyaratan dan kesesuaian tes ditampilkan di Table 9.1.

Tabel 9.1 Persyaratan Kualitas Software dan Klasifikasi Testing

Kategori	Persyaratan Kualitas	Sub Faktor	Klasifikasi Testing berdasarkan persyaratan
Operasional	Correctness (kebenaran)	▪ Akurasi dan kelengkapan output; Akurasi dan kelengkapan data ▪ Akurasi dan kelengkapan dokumentasi ▪ Ketersediaan (reaksi terhadap waktu) ▪ Pemrosesan data dan kebenaran perhitungan ▪ Koding dan Standarisasi dokumen	▪ Kebenaran testing output ▪ Testing dokumentasi ▪ Testing ketersediaan (reaksi waktu) ▪ Pemrosesan data dan testing kebenaran perhitungan
	Reliability (Ketahanan, kepercayaan)		Testing reliability
	Effisiensi		Testing tekanan (testing beban, testing waktu)
	Integritas		Testing keamanan sistem software
	Usability (dapat dipergunakan)	▪ Pelatihan yang dapat digunakan ▪ Operasional yang dapat digunakan	
Revision	Maintainability		Testing pemeliharaan

(Perbaikan)	(Pemeliharaan)		
	Fleksibel		Testing kelenturan
	Testability		Testing kemudah ditest
Transition (Peralihan)	Portability (Mudah dibawa)		
	Reusability (Penggunaan lagi)		
	Interoperability	▪ Interoperability dengan software dan peralatan lain	Testing interoperability software dan hardware

Aplikasi pengujian white box dan black box dalam pelaksanaan tes persyaratan telah mengungkapkan kelebihan dan kekurangan masing-masing konsep pengujian. Khususnya, sebagaimana telah tersirat, tes white box dari pengolahan data dan ketepatan perhitungan dapat digantikan dengan tes black box dari ketepatan output. Maintainability tes dapat diimplementasikan oleh white box dan tes black box, seperti temuan dari setiap konsep pengujian yang saling melengkapi. Pengujian persyaratan lainnya, bagaimanapun, karena karakteristik khusus mereka, dapat diterapkan sesuai dengan hanya satu atau konsep lainnya. Penerapan setiap konsep pengujian untuk berbagai factor persyaratan disajikan pada Tabel 9.2.

9.4. Testing White Box

Realisasi konsep pengujian white box membutuhkan verifikasi dari setiap laporan program dan komentar. Seperti yang ditunjukkan pada Tabel 9.2, pengujian white box memungkinkan kinerja pengolahan data dan tes perhitungan kebenaran, tes kualifikasi perangkat lunak, tes pemeliharaan dan tes reusabilitas.

Dalam rangka untuk melakukan pengolahan data dan tes kebenaran perhitungan ("pengujian kebenaran white box"), setiap operasi komputasi dalam urutan operasi yang dibuat oleh masing-masing kasus uji ("path") harus diperiksa. Jenis verifikasi memungkinkan kita untuk memutuskan apakah operasi pengolahan dan urutan mereka telah diprogram dengan benar untuk jalan tersebut, tetapi tidak untuk jalur lainnya. Beralih ke kualifikasi perangkat lunak, fokus di sini bergeser ke pemeriksaan kode perangkat lunak (termasuk komentar) sesuai dengan pengkodean standar dan instruksi kerja. Tes Maintainability menunjuk kepada fitur khusus, seperti yang dipasang untuk mendeteksi penyebab kegagalan, struktur modul yang mendukung adaptasi perangkat lunak dan perbaikan software, dll. Tes

reusability memeriksa peningkatan bahwa perangkat lunak yang digunakan kembali tergabung dalam paket dan adaptasi dilakukan untuk membuat sebagian dari perangkat lunak saat ini dapat digunakan kembali untuk paket perangkat lunak masa depan. Mengingat tujuan-tujuan dari tes SQA dan orientasi yang diadopsi oleh pengujian white box, bagian ini akan berhadapan dengan:

- Pengolahan data white box dan pengujian kebenaran perhitungan dan jumlah kasus uji yang diperlukan
- McCabe's cyclomatic complexity metrics
- Kinerja kualifikasi perangkat lunak dan tes reusabilitas
- Kelebihan dan kekurangan pengujian white box.

Tabel 9.2 Macam Kelas Testing Antara *White Box* dan *Black Box*

Klasifikasi Testing berdasarkan Persyaratan	White Box	Black Box
Testing kebenaran output		+
Testing dokumentasi		+
Testing ketersediaan (reaksi terhadap waktu)		+
Pemrosesan data dan testing kebenaran perhitungan	+	
Testing kualifikasi software	+	
Testing ketahanan		+
Testing tekanan (beban dan durasi)		+
Testing keamanan sistem software		+
Testing pelaksanaan pelatihan		+
Testing pelaksanaan operasional		+
Testing pemeliharaan	+	+
Testing fleksibilitas		+
Testing kemudahan dibawa (portability)		+
Testing penggunaan kembali	+	+
Testing interoperabiliti software		+
Testing interoperabiliti hardware		+

9.4.1. Pemrosesan Data dan Perhitungan Kebenaran Testing

Menerapkan konsep pengujian white box, yang didasarkan pada pemeriksaan pengolahan data untuk setiap kasus uji, segera menimbulkan kumpulan pertanyaan dari sejumlah kemungkinan besar jalur pemrosesan dan banyaknya baris kode. Dua pendekatan alternatif telah muncul.

■ "Path coverage" - untuk merencanakan pengujian kami untuk menutup semua jalan kemungkinan, di mana cakupan diukur dengan persentase jalan yang tertutup.

■ "Line coverage" - untuk merencanakan pengujian kami untuk menutup semua baris kode program, dimana cakupan diukur dengan persentase garis tertutup.

9.4.2. Kebenaran Testing dan Cakupan path

Jalur yang berbeda dalam sebuah modul software diciptakan oleh pilihan dalam pernyataan bersyarat, seperti IF-THEN-ELSE atau DO WHILE atau DO UNTIL. Pengujian Path dimotivasi oleh aspirasi untuk mencapai cakupan yang lengkap dari sebuah program dengan menguji semua path yang mungkin. Oleh karena itu, metric "path coverage" pengukuran kelengkapan jalur tes adalah didefinisikan sebagai persentase dari jalur program yang dieksekusi selama uji (diaktifkan dengan uji kasus yang termasuk dalam prosedur pengujian).

Sementara konsep jalur pengujian mengalir secara alami dari penerapan konsep pengujian white box, hal itu tidak praktis dalam banyak kasus karena luasnya sumber daya yang diperlukan untuk kinerjanya. Hanya sebagian besar biaya aplikasi ini dapat digambarkan dalam contoh berikut.

Mari kita menghitung jumlah jalur yang mungkin dibuat oleh modul sederhana yang berisi 10 pernyataan bersyarat, masing-masing memungkinkan untuk hanya dua pilihan (misalnya, IF-THEN-ALSO dan DO WHILE). Modul sederhana ini berisi 1024 jalan yang berbeda. Dengan kata lain, untuk mendapatkan cakupan path lengkap untuk modul ini (kemungkinan 25-50 baris kode) yang harus dipersiapkan adalah uji kasus minimal 1024, satu untuk setiap jalur yang mungkin. Perhitungan langsung dari jumlah kasus uji yang diperlukan untuk menguji sebuah paket software yang berisi 100 modul kompleksitas yang sama (sebanyak 102.400 kasus uji) siap menunjukkan tidak praktisnya penggunaan luas pengujian jalur. Oleh karena itu, penerapannya diarahkan terutama untuk modul perangkat lunak berisiko tinggi, di mana biaya kegagalan akibat kesalahan software sepenuhnya menjamin biaya pengujian jalur.

Situasi ini telah mendorong pengembangan alternatif namun memperlemah konsep cakupan dan cakupan line. Konsep line coverage membutuhkan jauh lebih sedikit uji kasus, tetapi, seperti yang dibayangkan, meninggalkan sebagian besar jalan yang mungkin belum teruji. Subyek dari line coverage dibahas berikutnya.

9.4.3. Kebenaran Testing dan Cakupan line

Konsep line coverage mensyaratkan bahwa, untuk line coverage yang penuh, setiap baris kode akan dieksekusi paling sedikit satu kali selama proses pengujian. Metrik line coverage untuk kelengkapan rencana pengujian-line ("pengujian jalur dasar") didefinisikan sebagai persentase dari garis yang dijalankan - yaitu, ditutupi - selama pengujian.

Untuk lebih memahami esensi pengujian jalur dasar dari sebuah program, referensi ke sebuah diagram alur (flowchart) dan program aliran grafik (flowgraph) dapat membantu. Dalam sebuah diagram alur, diamonds menyajikan pilihan yang tercakup oleh pernyataan bersyarat (keputusan), sedangkan persegi panjang atau persegi panjang suksesi merupakan bagian perangkat lunak yang menghubungkan persyaratan laporan keuangan tersebut. Dalam program aliran grafik, nodes menggambarkan bagian perangkat lunak dan dengan demikian menggantikan atu atau lebih aliran data persegi panjang. Pada bagian tepi menunjukkan urutan bagian perangkat lunak. Node memiliki dua atau lebih sisi yang merupakan pernyataan bersyarat. Contoh berikut menunjukkan diagram alir grafik dan aliran program untuk sebuah modul perangkat lunak Argometer yang menghitung tarif taksi.

Contoh - Imperial Taxi Service (ITS) Argometer

Imperial Taxi Service (ITS) melayani penumpang satu kali dan klien yang teratur (diidentifikasi dengan kartu taksi). Tarif taksi ITS untuk penumpang satu kali dihitung sebagai berikut:

(1) tarif Minimal: $ 2. Tarif ini mencakup jarak tempuh sampai dengan 1000 meter dan waktu tunggu (berhenti untuk lampu lalu lintas atau kemacetan lalu lintas, dll) hingga 3 menit.
(2) Untuk setiap 250 meter tambahan atau bagian dari itu: 25 sen.
(3) Untuk setiap 2 menit tambahan berhenti atau menunggu atau bagian daripadanya: 20 sen.
(4) Satu koper: tidak dipungut biaya; setiap koper tambahan: $ 1.
(5) Malam suplemen: 25%, efektif untuk perjalanan antara 21.00 dan 06.00.

Reguler klien berhak mendapatkan diskon 10% dan tidak dikenakan biaya suplemen malam.

Ketika merencanakan pengujian rencana jalur dasar dari modul Argometer baru, flow chart dan program aliran grafik untuk proses perhitungan ongkos taksi disusun. Setiap angka mewakili sebuah proses perhitungan yang meliputi lima keputusan, seperti yang ditunjukkan pada Gambar 9.3.

Sebuah tinjauan dari flowchart ITS dan program flowgraph menunjukkan perbedaan antara pengujian jalur dan pengujian jalur dasar sama baiknya seperti membandingkan kebutuhan pengujian dari path coverage dengan line coverage.

Seperti disebutkan di atas, path coverage yang lengkap mensyaratkan bahwa semua jalan yang mungkin dilaksanakan setidaknya sekali. Dalam diagram alur ITS (Gambar 9.3), 24 jalan yang berbeda mungkin ditunjukkan. Dengan kata lain, untuk mencapai path coverage lengkap dari modul perangkat lunak kami harus menyiapkan setidaknya 24 kasus uji, yang kami daftar dalam Tabel 9.3.

Table 9.3: The Imperial Taxi example – the full list of paths

No.	The path
1	1-2-3-5-6-8-9-11-12-17
2	1-2-3-5-6-8-9-11-13-14-15-17
3	1-2-3-5-6-8-9-11-13-14-16-17
4	1-2-3-5-6-8-10-11-17
5	1-2-3-5-6-8-10-11-13-14-15-17
6	1-2-3-5-6-8-10-11-13-14-16-17
7	1-2-3-5-7-8-9-11-12-17
8	1-2-3-5-7-8-9-11-13-14-15-17
9	1-2-3-5-7-8-9-11-13-14-16-17
10	1-2-3-5-7-8-10-11-12-17
11	1-2-3-5-7-8-10-11-13-14-15-17
12	1-2-3-5-7-8-10-11-13-14-16-17
13	1-2-4-5-6-8-9-11-12-17
14	1-2-4-5-6-8-9-11-13-14-15-17
15	1-2-4-5-6-8-9-11-13-14-16-17
16	1-2-4-5-6-8-10-11-12-17
17	1-2-4-5-6-8-10-11-13-14-15-17
18	1-2-4-5-6-8-10-11-13-14-16-17
19	1-2-4-5-7-8-9-11-12-17
20	1-2-4-5-7-8-9-11-13-14-15-17
21	1-2-4-5-7-8-9-11-13-14-16-17
22	1-2-4-5-7-8-10-11-12-17
23	1-2-4-5-7-8-10-11-13-14-15-17
24	1-2-4-5-7-8-10-11-13-14-16-17

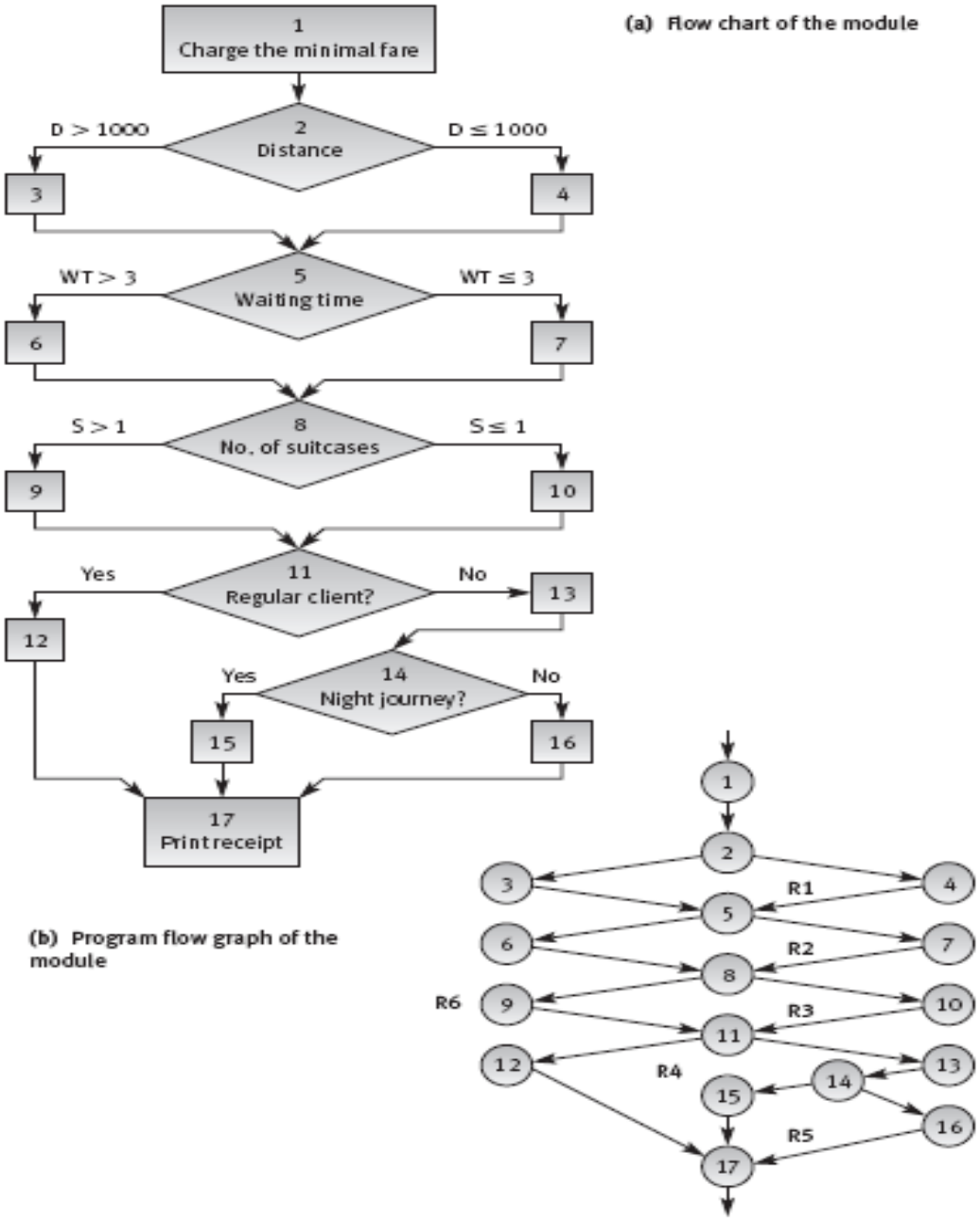


Figure 9.3: The ITS taxi fare calculation process – flow chart and program flow graph

Sebaliknya, grafik alur program memungkinkan kita untuk mengamati bahwa cakupan garis penuh dari modul software ITS dapat dicapai dengan memeriksa jumlah minimum jalur - total tiga - seperti tercantum pada Tabel 9.4.

Proporsi kasus uji yang diperlukan untuk menguji sistem dengan cakupan garis penuh dari tiga kasus uji (dengan menguji path dasar) versus cakupan path lengkap dari 24 kasus tes adalah 1:8! Rasio ini tumbuh pesat dengan kompleksitas program.

Dukungan untuk strategi pengujian jalur dasar ini disediakan oleh metrik cyclomatic kompleksitas McCabe, yang selain menjadi metrik kompleksitas software juga berfungsi untuk memberikan batas atas jumlah kasus uji yang diperlukan untuk line coverage yang lengkap.

9.4.4. Pengukuran Kompleksitas Siklomatik Mc Cabe

Metrik kompleksitas cyclomatic yang dikembangkan oleh McCabe (1976) mengukur kompleksitas program atau modul pada waktu yang sama seperti menentukan jumlah maksimum jalur independen yang diperlukan untuk mencapai cakupan garis penuh dari program tersebut. Ukuran ini didasarkan pada teori graph dan dengan demikian dihitung sesuai dengan karakteristik program seperti yang ditangkap oleh grafik aliran program.

Jalur independen didefinisikan dengan mengacu pada suksesi jalur independen akumulasi, yaitu: "Setiap jalan pada grafik alur program yang mencakup setidaknya satu sisi yang tidak termasuk dalam salah satu jalur independen yang dibentuknya".

Untuk menggambarkan definisi ini, mari kita simak sekali lagi untuk Gambar 9.3. Satu set dari jalur independen yang mencapai cakupan garis penuh dari program ini tercantum dalam Tabel 9.5.

Table 9.4: The Imperial Taxi example – the minimum number of paths

No.	The path
1	1-2-3-5-6-8-9-11-12-17
23	1-2-4-5-7-8-10-11-13-14-15-17
24	1-2-4-5-7-8-10-11-13-14-16-17

Table 9.5: The ITS example – the set of independent paths to achieve full coverage

Path no.	The path	Edges added by the path	Number of edges added by the path
1	1-2-3-5-6-8-9-11-12-17	1-2, 2-3, 3-5, 5-6, 6-8, 8-9, 9-11, 11-12, 12-17	9
2	1-2-3-5-6-8-9-11-13-14-15-17	11-13, 13-14, 14-15, 15-17	4
3	1-2-3-5-6-8-9-11-13-14-16-17	14-16, 16-17	2
4	1-2-4-5-7-8-10-11-13-14-15-17	2-4, 4-5, 5-7, 7-8, 8-10, 10-11	6

Sebagaimana disebutkan di atas, metric kompleksitas cyclomatic V(G) juga menentukan jumlah maksimum jalur independen yang dapat ditunjukkan dalam grafik aliran program.

Metrik kompleksitas cyclomatic (V (G)) dinyatakan dalam tiga cara yang berbeda, yang semuanya didasarkan pada grafik aliran program:

(1) $V(G) = R$

(2) $V(G) = E - N + 2$

(3) $V(G) = P + 1$

Dalam persamaan diatas, R adalah sejumlah daerah di grafik aliran program. Setiap daerah tertutup dalam grafik dianggap suatu daerah. Selain itu daerah sekitar grafik tidak tertutup oleh perhitungan sebagai satu daerah tambahan. E adalah jumlah edge pada grafik aliran program, N adalah jumlah node pada grafik aliran program, dan P adalah jumlah keputusan yang terdapat dalam grafik, diwakili oleh node memiliki lebih dari satu sisi.

Contoh

Menerapkan persamaan di atas sebagai contoh Argometer modul PERUSAHAAN dijelaskan di atas, kita dapat memperoleh nilai-nilai parameter di atas dari Gambar 9.3. Kami menemukan bahwa R = 6, E = 21, N 17 =, dan P = 5. Bila mensubstitusikan nilai-nilai ini ke dalam metrik formula kita mendapatkan:

(1) $V(G) = R = 6$

(2) $V(G) = E - N + 2 = 21 - 17 + 2 = 6$

(3) $V(G) = P + 1 = 5 + 1 = 6$

Hasil perhitungan metrik menunjukkan bahwa jumlah maksimum jalur independen dalam contoh adalah enam. Salah satu realisasi dari serangkaian maksimal dari enam jalur independen ditunjukkan pada Tabel 9.6.

Beberapa studi empiris dari hubungan antara metrik kompleksitas cyclomatic dan kualitas dan karakteristik testability telah dilakukan selama bertahun-tahun. Beberapa temuan dirangkum oleh Jones (1996) sebagai berikut: "studi empiris menunjukkan bahwa program-program dengan kompleksitas cyclomatic kurang dari 5 umumnya dianggap sederhana dan mudah dipahami. Kompleksitas cyclomatic

dari 10 atau kurang dianggap tidak terlalu sulit, jika 20 atau lebih, kompleksitas dianggap tinggi. Ketika nilai McCabe melebihi 50, perangkat lunak untuk tujuan praktis menjadi tak teruji "publikasi lain melaporkan tidak ada konfirmasi dari hubungan antara metrik kompleksitas cyclomatic dan kualitas perangkat lunak, atau bahwa hubungan yang ditemukan belum didukung secara statistik (Fenton, 1995, pp. 279-281)..

Table 9.6: The ITS example – the maximum set of independent paths

Path no.	The path	Edges added by the path	Number of edges added by the path
1	1-2-3-5-6-8-9-11-12-17	1-2, 2-3, 3-5, 5-6, 5-8, 8-9, 9-11, 11-12, 12-17	9
2	1-2-4-5-6-8-9-11-12-17	2-4, 4-5	2
3	1-2-3-5-7-8-9-11-12-17	5-7, 7-8	2
4	1-2-3-5-6-8-10-11-12-17	8-10, 10-11	2
5	1-2-3-5-6-8-9-11-13-14-15-17	11-13, 13-14, 14-15, 15-17	4
6	1-2-3-5-6-8-9-11-13-14-16-17	14-16, 16-17	2

9.4.5. Kualifikasi Software dan Testing Pemakaian Ulang

Software pengujian kualifikasi

Meskipun subjek kualifikasi telah dibahas dalam Bagian 7.3, topik itu tidak habis. Pengujian kualifikasi pengujian sangat penting untuk coding dalam pengembangan serta tahap pemeliharaan. Untuk review yang cepat, software yang memenuhi syarat dikodekan dan didokumentasikan sesuai dengan standar, prosedur dan instruksi kerja. Hal ini membuat lebih mudah bagi pemimpin tim untuk memeriksa perangkat lunak, bagi programmer pengganti untuk memahami kode dan melanjutkan tugas coding, dan bagi programmer pemelihara untuk memperbaiki bug dan / atau memperbarui atau mengubah program atas permintaan.

Pengujian kualifikasi software menengahkan apakah pengembangan perangkat lunak menanggapi positif atas pertanyaan tertentu yang mencerminkan satu set kriteria:

■ Apakah kode memenuhi struktur kode instruksi dan prosedur, seperti ukuran modul, aplikasi dari code yang telah digunakan, dll?

■ Apakah gaya pengkodean memenuhi prosedur gaya pengkodean?

■ Apakah dokumentasi program internal dan bagian “help” memenuhi prosedur gaya pengkodean?

Paket perangkat lunak khusus (yang disebut code auditors) sekarang dapat melakukan sebagian dari tes kualifikasi dengan mendaftar instances dari non-comformity untuk standard coding, prosedur dan instruksi kerja. Tes-tes lain terus beristirahat pada personil terlatih untuk eksekusi manual mereka.

Pengujian reusabilitas software

Reusabilitas software secara substansial mengurangi persyaratan sumber daya proyek dan meningkatkan kualitas sistem perangkat lunak baru. Dalam melakukannya, reusabilitas memperpendek masa pengembangan, yang dengan sendirinya menguntungkan organisasi pengembangan perangkat lunak. Pengujian Reusabilitas mendukung fungsi-fungsi dengan menentukan apakah kemasan dan dokumentasi program dan modul yang terdaftar untuk digunakan kembali sesuai dengan standar dan prosedur yang diminta untuk dimasukkan di perpustakaan penanggulangan perangkat lunak. Pengujian reusabilitas sebenarnya adalah salah satu alat untuk mendukung pertumbuhan software reuse.

9.4.6. Keuntungan dan Kerugian Testing White Box

Keuntungan utama pengujian whitebox adalah:

- Pemeriksaan kode per statement secara langsung memungkinkan penentuan kebenaran software seperti yang diungkapkan dalam jalur pengolahan, termasuk apakah algoritma telah didefinisikan pada program dengan benar.
- Memungkinkan kinerja follow-up cakupan baris (menerapkan paket software khusus) yang menyediakan penguji daftar baris kode yang belum dieksekusi. Penguji kemudian dapat memulai uji kasus yang mencakup baris kode tersebut.
- Mengetengahkan kualitas pekerjaan coding dan kepatuhan pada standar coding.

Kerugian utama dari pengujian whitebox adalah:

- Memanfaatkan sumber daya yang besar, jauh di atas yang dibutuhkan untuk pengujian blackbox dari paket perangkat lunak yang sama.
- Tidak mampu untuk menguji kinerja perangkat lunak dalam hal availability (waktu respon), reliability, load durability, dan kelas uji lainnya yang berhubungan dengan faktor operasi, revisi dan transisi.

Karakteristik pengujian whitebox membatasi penggunaan modul perangkat lunak dari risiko yang sangat tinggi dan biaya kegagalan yang tinggi, di mana hal tersebut sangat penting untuk mengidentifikasi dan memperbaiki sebanyak mungkin kesalahan-kesalahan dari perangkat lunak.

9.5. Testing Black Box

Pengujian blackbox memungkinkan kita untuk melakukan tes kebenaran output dan sebagian besar kelas-kelas tes seperti terlihat pada tabel 9.2. Selain dari tes output kebenaran (jika Anda siap untuk membayar biaya tambahan, ini dapat dilakukan dengan mengolah data whitebox dan menguji kebenaran perhitungan) dan maintainability tes (yang dapat dilakukan oleh tes whitebox), sebagian besar kelas pengujian lainnya unik untuk pengujian blackbox. Namun, karena karakteristik khusus dari setiap strategi pengujian dan kelas uji unik untuk pengujian whitebox, pengujian blackbox tidak bisa secara otomatis menggantikan pengujian whitebox.

Bagian ini akan berurusan dengan hal-hal berikut:

- Kelas ekivalen dan efeknya pada jumlah uji kasus yang diperlukan untuk pengujian kebenaran output.
- Kinerja metodologi untuk kelas lain dari tes blackbox.
- Keuntungan dan kerugian dari pengujian blackbox.

Untuk bahan tambahan pada pengujian blackbox, Beizer (1995) merupakan salah satu sumber utama yang tersedia.

9.5.1. Kelas Ekuivalen terhadap testing kebenaran output

Tes kebenaran output adalah, dalam banyak kasus, di antara tes yang mengkonsumsi sebagian besar sumber daya pengujian. Dalam kasus-kasus yang sering terjadi di mana tes kebenaran output ini dilakukan sendiri, mereka mengkonsumsi semua sumber daya pengujian. Implementasi kelas-kelas lain dari tes tergantung pada sifat dari produk perangkat lunak dan pengguna di masa depan serta pada prosedur pengembang dan keputusan.

Tes kebenaran output menerapkan konsep uji kasus. Meningkatkan pilihan uji kasus dapat dicapai dengan efisiensi penggunaan partisi kelas ekivalen, sebuah metode yang akan dibahas di sini.

Partisi kelas ekivalen adalah metode blackbox yang bertujuan untuk meningkatkan efisiensi pengujian dan, pada saat yang sama, meningkatkan cakupan kondisi kesalahan potensial. Kelas ekivalen (EC) adalah seperangkat nilai-nilai variabel input yang menghasilkan hasil output yang sama atau dari yang diproses secara identik. Batas EC ditentukan oleh nilai numerik atau abjad tunggal, sekelompok nilai numerik atau abjad, rentang nilai, dan sebagainya. suatu EC yang hanya berisi pernyataan valid didefinisikan sebagai " valid EC", sedangkan sebuah EC yang hanya berisi pernyataan tidak valid

didefinisikan sebagai " invalid EC". dalam kasus di mana sebuah input program disediakan oleh beberapa variabel, EC valid dan tidak valid harus didefinisikan untuk setiap variabel.

Berdasarkan pada metode partisi kelas ekivalen, uji kasus didefinisikan sehingga setiap EC valid dan setiap EC tidak valid dimasukkan dalam setidaknya satu kasus uji. Uji kasus didefinisikan secara terpisah untuk ECs valid dan tidak valid. Dalam mendefinisikan kasus uji untuk ECs valid, kami mencoba untuk mencakup sebanyak mungkin EC "baru" (yaitu, kelas yang tidak termasuk dalam salah satu uji kasus sebelumnya) dalam kasus pengujian yang sama. Uji kasus ditambahkan selama masih ada EC yang belum tercakup. Sebagai hasil dari proses ini, jumlah uji kasus dibutuhkan untuk mencakup EC valid yang sama dan dalam banyak kasus jauh di bawah jumlah EC valid. Dicatat bahwa dalam mendefinisikan EC tidak valid, kami menetapkan satu kasus uji untuk setiap EC tidak valid yang "baru", karena hanya satu EC tidak valid yang dapat dimasukkan dalam uji kasus. Uji kasus yang mencakup lebih dari satu EC tidak valid tidak memungkinkan penguji untuk membedakan antara reaksi terpisah program untuk masing-masing EC tidak valid. maka, jumlah kasus uji yang diperlukan untuk EC tidak valid sama dengan jumlah EC tidak valid.

Dibandingkan dengan penggunaan sampel acak dari uji kasus, kelas ekivalen menghemat sumber daya pengujian karena mereka menghilangkan duplikasi uji kasus yang ditetapkan untuk masing-masing EC. Utamanya, karena metode kelas ekivalen adalah metode blackbox, partisi kelas ekivalen didasarkan pada dokumentasi spesifikasi perangkat lunak, bukan pada kode. Pembangunan sistematis dari kelas ekivalen untuk variabel input sebuah program meningkatkan cakupan kondisi yang valid dan kesalahan yang mungkin timbul dari input dan dengan demikian meningkatkan efektivitas rencana pengujian itu. Perbaikan lebih lanjut uji efektivitas dan efisiensi dicapai dengan tes nilai batas EC, topik kita uraikan berikutnya.

9.5.2. Kelas Testing Faktor Operasional Lain

Pengujian dokumentasi, meskipun diabaikan dalam banyak kasus, seharusnya dianggap sama pentingnya dengan pengujian kode atau dokumen inspeksi desain. User manual dan programmer manual yang salah dapat menyebabkan kesalahan selama operasi program dan pemeliharaan yang mungkin timbul kerusakan setara keparahan yang disebabkan oleh bug perangkat lunak.

Komponen umum dari dokumentasi, yang disediakan oleh pengembang, adalah:

- Deskripsi fungsional dari sistem perangkat lunak. Ikhtisar ini memungkinkan pengguna potensial untuk memutuskan apakah sistem ini cocok untuk kebutuhannya atau tidak.

- Petunjuk instalasi. Dokumen ini meliputi penjelasan rinci dari proses instalasi dan persyaratan perangkat keras serta instruksi untuk berinteraksi dengan peralatan dan dengan paket perangkat lunak lain - jika antarmuka tersebut merupakan bagian dari spesifikasi sistem. Pada paket perangkat lunak komersial (Cots software), instalasi biasanya mencakup juga instruksi kustomisasi.
- Panduan pengguna. Petunjuk "bagaimana untuk memulai", instruksi rinci untuk menerapkan berbagai fungsi sistem, petunjuk pemulihan untuk kesalahan umum operator dan kesalahan sistem semua tercakup dalam buku petunjuk ini. Dalam banyak kasus, petunjuk pengguna diberikan sebagai panduan bantuan yang terkomputerisasi.
- Panduan programmer. Dokumentasi jenis ini diberikan untuk perangkat lunak custom made. itu termasuk informasi yang diperlukan untuk mempertahankan sistem (koreksi bug, adaptasi terhadap perubahan kebutuhan dan perbaikan perangkat lunak), struktur program, deskripsi program, termasuk algoritma logika, dan sebagainya. Pengguna dan pelanggan perangkat lunak COTS tidak memerlukan sebuah panduan programmer seperti mereka tidak melakukan pemeliharaan sendiri.

Dokumen rencana pengujian seharusnya mencakup tiga komponen berikut:

- Memeriksa kelengkapan dokumen. Berdasarkan spesifikasi kebutuhan dan laporan desain detail, tujuannya adalah untuk memeriksa apakah semua dokumen yang diperlukan telah diselesaikan seperti yang ditetapkan dan sebagaimana dimaksudkan oleh perancang.
- Memeriksa kebenaran dokumen. Uji kebenaran ditentukan apakah petunjuk yang tercantum dalam dokumen pengguna sudah benar. Pelaksanaan uji kebenaran membutuhkan merancang sebuah file ujian dengan cara yang mirip dengan metodologi tes ketelitian output yang dibahas pada bagian 9.5.1.
- Document style dan editing inspection. Mengacu pada kejelasan dokumen dan kesepakatannya dengan standar dokumenasi dalam kasus ini persyaratan ditentukan dalam kontrak.

Availability Test

Availability didefinisikan sebagai reaksi waktu - waktu yang dibutuhkan untuk memperoleh informasi yang diminta atau waktu yang diperlukan untuk firmware yang terpasang pada peralatan terkomputerisasi untuk bereaksi. Availability adalah yang paling penting dalam aplikasi on-line sistem informasi yang sering digunakan. Kegagalan firmware perangkat lunak untuk memenuhi persyaratan availability (yaitu, waktu reaksi terbelakang) dapat membuat peralatan berguna.

Relatif sulit untuk menguji availability, terutama untuk sistem informasi direncanakan untuk melayani populasi besar pengguna, dan untuk sistem real time direncanakan untuk menangani peristiwa frekuensi tinggi. Kesulitan ini berasal dari kebutuhan untuk melaksanakan tes di bawah beban operasi normal maupun dalam kondisi beban maksimal yang ditetapkan dalam spesifikasi kebutuhan. Perlu dicatat bahwa persyaratan availability untuk beban kerja reguler dan maksimal biasanya berbeda. Karakteristik

yang dibutuhkan oleh lingkungan pengujian ketersediaan mendukung gabungan dari kelas uji dengan load test (stress test) dan melakukan tes gabungan yang terkomputerisasi. (Untuk diskusi tentang load test, lihat di bawah ini).

Reliability Test

Persyaratan keandalan sistem perangkat lunak berkaitan dengan fitur yang dapat diterjemahkan sebagai peristiwa yang terjadi dari waktu ke waktu, seperti waktu rata-rata antara kegagalan (misalnya, 500 jam), waktu rata-rata untuk pemulihan setelah kegagalan sistem (misalnya, 15 menit), atau downtime rata-rata per bulan (misalnya, 30 menit per bulan). persyaratan keandalan harus berlaku selama operasi penuh kapasitas reguler dari sistem. Perlu dicatat bahwa sebagai tambahan dalam faktor perangkat lunak, reliability test juga berhubungan dengan perangkat keras, sistem operasi, dan efek sistem komunikasi data.

Seperti availability test, reliability test sangat sulit dilakukan karena membutuhkan pengoperasian berbagai aplikasi perangkat lunak yang dilakukan dalam kondisi beban kerja reguler. Agar praktis, tugas-tugas seperti itu harus dilakukan hanya setelah simulasi komputer telah dijalankan untuk mendapatkan nilai rata-rata, dan hanya setelah sistem selesai. Sehubungan dengan sumber daya, kendala utama dalam pelaksanaan pengujian jenis ini adalah ruang lingkup sumber daya yang dibutuhkan sangat luas, seperti pengujian dapat terus berlangsung selama ratusan jam dan sebuah file uji kasus komprehensif harus dibangun.

Statistik reliability test menawarkan pilihan yang jauh lebih murah dan lebih cepat untuk penilaian keandalan berdasarkan model statistik. literatur yang tersedia pada subjek, hanya beberapa sumber utama yaitu Myers (1976), Musa et al. (1990) dan Musa (1998). Namun, meskipun digunakan secara luas dan bermanfaat praktis, statistik reliability test telah menjadi sasaran kritik sejak kemunculan mereka. Isu utama yang diperdebatkan adalah sejauh mana model statistik mewakili kehidupan nyata sistem operasi software.

Stress Test

Kelas dari stress test digolongkan dalam dua tipe uji utama: load test dan durability test. Dimungkinkan untuk melakukan tes ini hanya setelah penyelesaian perangkat lunak. Durability test, bagaimanapun, secara umum dapat dilakukan hanya setelah firmware atau sistem informasi perangkat lunak telah terinstal dan siap untuk pengujian.

Stress Test : Load Test

Load test berhubungan dengan kinerja fungsional dari sistem di bawah beban operasional maksimal: transaksi maksimal per menit, hits per menit untuk sebuah situs internet dan sejenisnya. Load test, yang biasanya dilakukan untuk beban lebih tinggi dari yang ditunjukkan spesifikasi kebutuhan, adalah sesuatu yang sangat penting untuk sistem software yang direncanakan untuk melayani populasi besar pengguna

secara bersamaan. Di sebagian besar sistem kerja perangkat lunak, angka beban maksimal menggabungkan beberapa jenis transaksi. Karena variabilitasnya, cara terbaik untuk menjelaskan proses adalah melalui contoh.

Contoh

“Music in the Air”, sebuah jaringan toko musik, menjalankan sebuah layanan di internet yang mendaftar permintaan-permintaan untuk penawaran harga dan pesanan. Pada hari kerja, tingkat rata-rata hits pelanggan adalah 10 per menit untuk pemesanan dan 20 per menit untuk penawaran harga. Beban maksimum terekam pada hari Sabtu adalah 30 per menit untuk pesanan dan 100 per menit untuk penawaran harga. Beban maksimum yang ditetapkan dalam spesifikasi perangkat lunak, yang membawa pertumbuhan di masa depan ke dalam rekening, adalah 60 per menit untuk pesanan dan 200 per menit untuk penawaran harga. Beban untuk program yang akan diuji adalah 75 per menit untuk pesanan dan 250 per menit untuk penawaran harga. Jadi, ini menjelaskan bagaimana beban uji dipilih sebagai contoh.

Manual kinerja dari load test tidak praktis untuk sebagian besar sistem perangkat lunak, dan karena itu dilakukan dengan tes komputer berdasarkan simulasi komprehensif beban tinggi, lagi mirip dengan prosedur yang disesuaikan untuk pengujian ketersediaan. Beban simulasi ini memungkinkan kita untuk mengukur waktu reaksi yang diharapkan sebagai fungsi dari sejumlah beban. Sehingga memungkinkan kita untuk memastikan apakah upgrade yang diperlukan dan perubahan yang harus dilakukan untuk memungkinkan sistem perangkat lunak untuk memenuhi persyaratan yang direncanakan. Untuk lebih lanjut tentang tes beban terkomputerisasi, lihat bagian 10.3 dalam bab berikutnya yang berkaitan dengan pengujian perangkat lunak komputerisasi.

Stress Test : Durability Test

Durability test dilakukan dalam kondisi operasi fisik yang ekstrim seperti suhu tinggi, kelembaban, dan mengemudi dengan kecepatan tinggi sepanjang jalan tidak beraspal, seperti yang dijelaskan dalam spesifikasi persyaratan ketahanan. Maka, durability test ini biasanya diperlukan untuk firmware real-time terintegrasi ke dalam sistem seperti sistem senjata, kendaraan transportasi jarak jauh, dan peralatan meteorologi. Masalah ketahanan untuk firmware termasuk tanggapan firmware untuk efek iklim seperti suhu panas dan dingin yang ekstrim, debu, gundukan jalan, dan kegagalan operasi ekstrim akibat dari gangguan listrik tiba-tiba, tegangan "melompat" pada pasokan listrik, komunikasi terputus secara tiba-tiba, dan sebagainya. Tes ketahanan sistem informasi software fokus pada kegagalan operasi akibat kegagalan listrik dan terputusnya komunikasi secara tiba-tiba.

Uji Sistem Keamanan Perangkat Lunak

Komponen keamanan perangkat lunak dari sistem perangkat lunak bertujuan untuk mencegah akses yang tidak sah ke sistem atau bagian darinya, deteksi akses yang tidak sah dan kegiatan yang dilakukan oleh penetrasi, dan pemulihan kerusakan yang disebabkan oleh penetrasi kasus yang tidak sah.

Masalah-masalah keamanan utama yang ditangani oleh tes-tes ini adalah :

- Kontrol akses, dimana kebutuhan yang lazim adalah untuk kontrol akses multi level (biasanya dengan mekanisme password). Hal khusus di sini adalah sistem firewall yang mencegah akses tanpa izin ke situs internet.
- Backup database dan file perangkat lunak dan pemulihan dalam kasus kegagalan sistem.
- Pencatatan transaksi, penggunaan sistem, uji coba akses, dan sebagainya.

Tantangan untuk menciptakan dan membobol sistem keamanan telah menghasilkan sebuah merek khusus dari kejahatan, yaitu hacker. Seringnya sangat muda, penggemar ini menemukan kesenangan yang teramat sangat terlebih dahulu, dan terutama dengan membobol sistem computer yang kompleks, kadang-kadang disertai dengan gangguan sistem, atau penciptaan virus yang melumpuhkan orang lain. Keberhasilan mereka telah mengejutkan dalam beberapa kasus (misalnya bank nasional, sistem keamanan militer AS, dll), dan memalukan pada tingkat yang sama. Sebuah "hasil" dari kesuksesan mereka adalah bahwa hal itu tidak lagi langka untuk menemukan hacker diundang untuk bergabung dengan tim tester, khususnya untuk sistem perangkat lunak dengan persyaratan keamanan yang tinggi.

Training Usability Test

Ketika sejumlah besar pengguna terlibat dalam sebuah sistem operasi, kebutuhan pelatihan kegunaan ditambahkan ke agenda pengujian. Ruang lingkup kegunaan pelatihan ditentukan oleh sumber daya yang dibutuhkan untuk melatih karyawan baru, dengan kata lain, berapa jam pelatihan yang dibutuhkan karyawan baru untuk mencapai tingkat yang ditetapkan dikenalkan dengan sistem atau untuk mencapai tingkat produksi yang ditetapkan per jam rincian tes ini, seperti yang lain, didasarkan pada karakteristik sistem tetapi, yang lebih penting di sini, di karakteristik karyawan. hasil tes harus menginspirasi rencana canggih kursus pelatihan dan tindak lanjut serta arah yang lebih baik untuk sistem operasi software.

Operational Usability Test

Fokus dari kelas uji ini adalah produktivitas operator, yaitu aspek-aspek dari sistem yang mempengaruhi kinerja secara teratur dicapai oleh operator sistem, atau yang diterapkan terutama untuk sistem informasi yang melayani banyak pengguna. tes ini sangat penting dalam kasus di mana cara kerja sistem dapat mempengaruhi substansial produktivitas penggunanya.

Pelaksanaan kelas ini terutama berkaitan dengan pengujian, produktivitas kuantitatif, dan kualitatif. Secara alami, alami aspek-aspek ini sangat penting bagi sistem yang berfungsi sebagai alat kejuruan utama untuk sekelompok besar pengguna.

Tes kegunaan operasional dapat dilakukan secara manual dengan menggunakan studi waktu. Selain data produktivitas, tes manual menyediakan beberapa wawasan ke dalam alasan untuk (tinggi rendah) tingkat kinerja dan memulai ide untuk perbaikan. Catatan kinerja yang akurat dapat dicapai dengan otomatis menindaklanjuti software yang mencatat semua aktivitas pengguna di seluruh shift mereka. Paket perangkat lunak jenis ini menyediakan statistic kinerja dan perbandingan variable yang berbeda, seperti aktivitas tertentu, jangka waktu, dan industri.

Diskusi komprehensif masalah pengujian kegunaan dan contoh rinci dapat ditemukan pada Rubin (1994).

9.5.3. Kelas Testing Faktor Perbaikan

Revisi mudah dari perangkat lunak adalah faktor fundamental memastikan paket perangkat lunak yang sukses, layanan panjang dan penjualan yang sukses untuk populasi pengguna yang lebih besar. terkait dengan fitur ini revisi kelas pengujian dibahas dalam bagian ini:

- Maintainability test
- Flexibility test
- Testability test

Maintainability test

Pentingnya pemeliharaan dan maintainability suatu software tidak pernah berlebihan. Mempertimbangkan fakta bahwa fungsi-fungsi tersebut membutuhkan lebih dari 60% dari total sumber daya perancangan dan pemrograman yang ditanamkan dalam sebuah sistem software di seluruh siklus hidupnya (Pressman, 2000). Meskipun perkiraan dari pembagian sumber daya untuk pemeliharaan berubah-ubah - lebih dari 50% seperti yang dilaporkan oleh Lientz dan Swanson (1980) dan 65-75% seperti yang dilaporkan oleh McKee (1984) - pentingnya pembagian sumber daya untuk pemeliharaan tak dapat disangkal.

Maintainability test sebagian besar terkait pada beberapa poin berikut ini:

1) Struktur dari sistem patuh pada standar dan prosedur pengembangan diterapkan pada komponen khusus untuk mendukung aktivitas pemeliharaan yang akan datang, termasuk struktur yang modular dari modul yang berisikan dirinya sendiri dan ukuran modul.
2) Buku petunjuk untuk programmer disiapkan berdasarkan pada standar dokumentasi yang disetujui dan menyediakan dokumentasi sistem yang lengkap.
3) Dokumentasi bagian dalam yang tergabung dalam kode software disiapkan berdasarkan pada prosedur pemrograman dan konvensi dan mencakup semua persyaratan dokumentasi sistem.

Tes kualifikasi software adalah alat penjamin kualitas software yang dipilih untuk pengecekan ketaatan pada persyaratan kemampuan memelihara seperti pada point (1) di atas (lihat bagian 9.4.4). pengujian

ketaatan terhadap persyaratan dari point (2) dan (3) jatuh pada bidang dari dokumentasi percobaan programmer dan ditampilkan kecuali termasuk dalam tes dokumentasi user (lihat bagian 9.5.2)

Flexibility test

Fleksibilitas sistem perangkat lunak mengacu pada kemampuan sistem, berdasarkan karakteristik wilayah struktural dan pemrograman. Faktor-faktor ini berpengaruh secara signifikan terhadap upaya yang diperlukan untuk mengadaptasi perangkat lunak untuk berbagai kebutuhan pelanggan, serta untuk memperkenalkan perubahan diprakarsai oleh pelanggan dan tim pemeliharaan untuk tujuan meningkatkan fungsionalitas sistem.

Tes fleksibilitas ini dimaksudkan untuk menguji karakteristik perangkat lunak yang mendukung fleksibilitas, seperti struktur modular yang memadai dan penerapan opsi parameter untuk menyediakan berbagai aplikasi yang luas.

Testability test

Persyaratan testability berurusan dengan kemudahan pengujian sistem perangkat lunak. Demikian, testability sini berkaitan dengan penambahan fitur khusus dalam program yang membantu para penguji dalam pekerjaan mereka, seperti kemungkinan untuk mendapatkan hasil untuk pos pemeriksaan tertentu dan file-file log standar. walaupun sering terabaikan, fitur ini khusus dukungan pengujian harus ditentukan dalam dokumen persyaratan sebagai bagian integral persyaratan fungsional perangkat lunak.

Tujuan lain dari kesepakatan pengujian dengan aplikasi peralatan diagnostik diterapkan untuk analisis kinerja sistem dan laporan kegagalan apapun yang ditemukan. Beberapa fitur semacam ini sudah diaktifkan secara otomatis ketika memulai mengemas perangkat lunak atau selama operasi rutin dan laporan jika kondisi jaminan alarm muncul. Fitur lain jenis ini mungkin diaktifkan oleh operator atau teknisi pemeliharaan. Prose pengujian adalah hal utama yang sangat penting untuk mendukung ruang kontrol sistem operasi yang besar (misalnya, listrik tanaman) dan untuk tim pemeliharaan, terutama berkenaan atau berhubungan dengan diagnosis kegagalan. Aplikasi pendukung pemeliharaan pada jenis ini mungkin diaktifkan di lokasi pelanggan dan atau di beberapa pusat bantuan yang tersedia pada wilayah pelanggan.

Testability tes akan dilakukan untuk aplikasi dari kedua jenis, sebagaimana tercatat dalam spesifikasi kebutuhan. Pengujian harus terkait terutama dengan aspek kebenaran, dokumentasi dan ketersediaan, sebagaimana telah dibahas sebelumnya.

9.5.4. Kelas Testing Faktor Transisi

Karakteristik perangkat lunak yang diperlukan untuk pengoperasian yaitu antara lain,perangkat lunak dengan penyesuaian kecil,di lingkungan yang berbeda, dan yang diperlukan untuk memasukkan kembali modul atau untuk mengizinkan interfacing dengan paket perangkat lunak lain selama interfacing itu dibutuhkan pada fitur transisi dari sistem perangkat lunak, khususnya untuk paket perangkat lunak komersial yang ditujukan untuk berbagai pelanggan. Oleh karena itu,kelas pengujian berikut, yang dibahas dalam bagian ini, harus diterapkan antara lain:

(1) tes Portabilitas

(2) tes Reusability(penggunaan kembali)

(3) Pengujian untuk persyaratan kemampuan pengoperasionalisasian (interoperabilitas) :

■ Perangkat Lunak interfacing tes

■ Peralatan interfacing tes.

Tes Portabilitas

Persyaratan Portabilitas menentukan lingkungan (atau kondisi lingkungan) di mana perangkat lunak sistem harus beroperasi: operasi sistem, perangkat keras dan standart peralatan komunikasi , antara lain variabel. Tes portabilitas untuk akan melakukan verifikasi, validasi dan pada pengujian faktor ini juga memperkirakan sumber daya yang diperlukan untuk transfer system software lingkungan yang berbeda.

Tes Reusability

Reusability(penggunaan kembali) mendefinisikan bagian mana program (modul, integrasi dan sejenisnya) harus dikembangkan untuk digunakan kembali di masa depan dalam pengembangan proyek perangkat lunak lain, baik yang sudah direncanakan atau tidak. Bagian ini harus dikembangkan,dikemas dan didokumentasikan menurut prosedur penggunaan kembali libarary perangkat lunak .Persyaratan Reusability adalah merupakan susatu hal penting atau khusus untuk proyek perangkat lunak yang berorientasi obyek. Oleh karena itu pengujian ini dirancang untuk menguji apakah standart reusabilitas memang dipatuhi.

Tes interoperabilitas Perangkat Lunak

Interoperabilitas perangkat lunak behubungan dengan kemampuan perangkat lunak untuk melakukan interfacing(penggabungan) peralatan dan paket perangkat lunak lain, untuk memungkinkan mereka beroperasi bersama-sama sebagai satu sistem komputerisasi yang kompleks. Daftar persyaratan delineates peralatan yang spesifik dan / atau perangkat lunak antarmuka yang akan diuji, sertaberlaku transfer data dan standart interfacing. Sebagian pertumbuhan komersial over-the-counter (COTS) paket perangkat lunak serta paket perangkat lunak yang dibuat khusus sekarang diharuskan untuk memiliki kemampuan interoperabilitas, yaitu, untuk menampilkan kapasitas untuk menerima masukan dari firmware peralatan dan / atau system software dan / atau mengirimkan output ke firmware lain dan system perangkat lunak. Kemampuan perangkat lunak ini dilakukan berdasarkan pada standart transfer data rigid, interoperabilitas internasional dan global atau industri yang berorientasi pada standar interoperabilitas, dan telah diuji dengan sesuai.

Tes interoperabilitas peralatan

Interoperabilitas peralatan berhubungan dengan interfacing peralatan firmware dengan unit peralatan lainnya dan / atau paket perangkat lunak, di mana daftar kebutuhan antarmuka yang ditentukan, termasuk dengan standar interfacing. Relevan tes harus memeriksa pelaksanaan interoperabilitas seluruh peralatan yang dibutuhkan selama persyaratan sistem.

9.5.5. Keuntungan dan Kerugian Testing Black Box

Keuntungan utama dari balck box tes adalah:

- Black box tes memungkinkan kita untuk melaksanakan sebagian besar kelas pengujian, yang sebagian besar dapat diimplementasikan hanya dengan tes kotak hitam. Pengujian kelas unik pada black box tes,adalah pengujian kinerja sistem tes seperti tes viral dan tes ketersediaan,pengujian ini sangatlah penting.
- Untuk kelas pengujian yang bisa dilakukan oleh keduanya yaitu white box dan black box tes, black box tes memerlukan sumber daya lebih sedikit daripada yang dibutuhkan untuk white box tes dari paket perangkat lunak yang sama.

Kelemahan utama black box tes adalah:

- Kemungkinan bahwa kebetulan agregasi dari beberapa kesalahan akan menghasilkan respon yang benar untuk kasus pengujian, dan mencegah deteksi kesalahan. Dengan kata lain, black box tes tidak mudah mengidentifikasi kesalahan dari kasus-kasus yang berlawanan satu sama lain yang untuk sengaja menghasilkan output yang benar.
- Tidak adanya kontrol cakupan baris. Dalam kasus di mana penguji black box berkeinginan untuk meningkatkan cakupan garis, tidak ada cara mudah untuk menentukan parameter dari pengujian kasus yang diperlukan untuk meningkatkan cakupan baris tersebut. Akibatnya, black box tes mungkin tidak menjalankan sebagian besar baris dari kode, yang tidak tercakup oleh satu set pengujian kasus.
- Kemungkinan pengujian dari kualitas pengkodean dan kepatuhan yang ketat untuk standar pengcodingan.

9.6. Kesimpulan

(1) Menjelaskan tentang tujuan pengujian.

Seseorang harus dapat membedakan antara tujuan pengujian langsung dan tidak langsung.

Tujuan dari pengujian langsung:

- Untuk mengidentifikasi dan mengungkapkan kesalahan sebanyak mungkin dalam pengujian perangkat lunak.
- Untuk membawa pengujian perangkat lunak ke tingkat kualitas yang dapat diterima
- Untuk menunjukkan pengujian yang dibutuhkan dengan cara yang efisien dan efektif, dengan melihat anggaran dan keterbatasan penjadwalan .

Tujuan tidak langsung:

- Untuk pasokan catatan kesalahan perangkat lunak yang akan digunakan untuk pencegahan kesalahan.

(2) Diskusikan perbedaan antara berbagai strategi pengujian, keuntungan dan kekurangannya.

Pada dasarnya ada dua strategi pengujian:

■ "Big bang testing": tes perangkat lunak secara keseluruhan, setelah paket selesai tersedia.

■ "Incremental testing(pengujian tambahan)": tes perangkat lunak sedikit demi sedikit - modul

perangkat lunak yang diuji karena sudah selesai diuji (unit test), diikuti oleh kelompok modul yang terdiri atas modul terintegrasi yang telah diuji dengan modul baru yang telah selesai diuji (tes integrasi).Setelah seluruh paket selesai, maka diuji secara keseluruhan (uji sistem).

Ada dua strategi dasar pengujian incremental: bottom-up dan top-down. Dalam pengujian top-down, modul pertama yang diuji adalah modul utama, modul tingkat tertinggi dalam

struktur perangkat lunak; modul terakhir yang akan diuji adalah modul yang berada pada tingkat terendah.Dalam pengujian bottom-up, urutan pengujian akan dibalik: modul tingkat terendah diuji pertama, dengan modul utama diuji terakhir.

■ Big bang vs pengujian tambahan.Karena program ini sangat kecil dan sederhana, menerapkan strategi pengujian "big bang" menyajikan kerugian yang parah.Identifikasi kesalahan pada seluruh paket perangkat lunak ketika dianggap sebagai salah satu "Unit" sangat sulit dan, meskipun sumber daya yang diinvestasikan besar , sangat tidak efektif. Selain itu, melakukan koreksi yang sempurna dari kesalahan dalam konteks ini adalah sering melelahkan. Jelas, perkiraan sumber daya pengujian yang diperlukan dan jadwal pengujian cenderung agak kabur. Sebaliknya, manfaat pengujian tambahan, karena dilakukan pada unit perangkat lunak yang relatif kecil, menghasilkan persentase pengidentifikasian kesalahan lebih tinggi dan mempunyai fasilitas koreksi atas kesalahan tersebut. Sebagai akibatnya, secara umum diterima bahwa tes tambahan(incremental testing) harus lebih disukai.

■ bottom-up vs top-down. Keuntungan utama dari strategi bottom-up adalah mempunyai kemudahan yang relatif dalam kinerjanya, sedangkan kekurangan utamanya adalah kelambatan dari tahap di mana dimungkinkan untuk mengamati program secara keseluruhan. Keuntungan utama strategi top-down adalah tahap awal di mana dimungkinkan untuk menunjukkan program secara keseluruhan, suatu kondisi yang mendukung identifikasi kesalahanawal dari analisis dan desain. Kelemahan utama dari pendekatan ini adalah kesulitan perbandingan kinerjanya.

(3) Jelaskan konsep white box tes dan black box tes, dan mendiskusikan kelebihan dan kekurangannya.

■ Black box testing mengidentifikasi bug-bug hanya berdasarkan pada kesalahan fungsi perangkat lunak

sebagaimana terungkap dari outputnya, sementara itu black box tes mengabaikan perhitungan dari jalur internal yang dilakukan oleh perangkat lunak.

■ White box tes meneliti perhitungan dari jalur internalyang bertujuan untuk mengidentifikasi bug.

Keuntungan utama dari black box tes adalah:

■Tes ini memungkinkan penguji untuk melakukan hampir semua kelas pengujian.

■Untuk kelas pengujian yang dapat dilakukan oleh kedua tes yaitu white box dan black box, black box testing membutuhkan sumber daya yang jauh lebih sedikit.

Kelemahan utama dari black box tes adalah:

■ Tes ini memungkinkan untuk mengidentifikasi kesalahan yang kebetulan sebagai benar.

■ Tes ini tidak memiliki kontrol cakupan baris.

■ Tes ini tidak memiliki kemungkinan untuk menguji kualitas kerja coding.

Keuntungan utama dari white box tes adalah:

■ Tes ini memungkinkan pemeriksaan langsung jalan pengolahan dan algoritma.

■Tes ini menyediakan tindak lanjut yang mengirimkan cakupan baris dari baris kode yang belum tereksekusi.

■ Tes ini memiliki kemampuan untuk menguji kealitas kerja koding.

Kelemahan utama white box tes adalah:

■ Tes ini membutuhkan sumber daya yang luas, jauh di atas yang dibutuhkan oleh balck box tes.

■ Tes ini tidak dapat memeriksa performa perangkat lunak dalam hal ketersediaan, keandalan, stres, dll

(4) Tentukan jangkauan cakupan jalur vs cakupan baris.

"Cakupan jalur" didefinisikan sebagai persentase jalur kemungkinan dari diaktifkannya pengolahan software oleh uji kasus. "Cakupan baris" didefinisikan sebagai persentase dari pemeriksaan baris kode yang dieksekusi selama pengujian. Konsep dari pengujian jalur dan cakupan baris berlaku untuk

memperkirakan cakupan white box tes saja. Dalam kebanyakan kasus, pencapaian cakupan jalur secara penuh tidak praktis,karena ruang lingkup sumber daya yang diperlukan untuk pelaksanaannya.

(5) Jelaskan berbagai jenis tes kotak hitam.

Black box tes diklasifikasikan ke dalam tiga kelompok:

- Kelas faktor pengujian operasi
- Kelas faktor pengujian revisi
- Kelas faktor pengujian transisi.

Kelas faktor pengujian Operasi meliputi:

(1) Uji kebenaran Output - dalam banyak kasus, kelas tes yang memakan sumber daya terbesar dalam pengujian adalah uji kebenaran output.

(2) Tes Dokumentasi (untuk kebenaran)

(3) Ketersediaan (waktu reaksi) tes (untuk kebenaran)

(4) Tes Keandalan

(5) Tes stress (tes viral, tes daya tahan) (untuk efisiensi)

(6) Pengujian sistem keamanan software

(7) Tes kegunaan Pelatihan (untuk kegunaan)

(8) Tes kegunaan Operasional (untuk kegunaan).

Kelas faktor pengujian Revisi meliputi:

(1) Tes Maintainability

(2) Tes Fleksibilitas

(3) Tes Testability .

Faktor kelas Transisi pengujian meliputi:

(1) Tes Portabilitas

(2) Tes Reusability

(3)Tes interfacing perangkat lunak (untuk interoperabilitas)

(4) Tes peralatan interfacing (untuk interoperabilitas).

Review pertanyaan

9.1 Tidak beberapa industry perangkat lunak profesional mempertahankan bahwa tujuan utama daripengujian perangkat lunakn adalah "untuk membuktikan bahwa paket perangkat lunak sudah siap".

(1) Jelaskan dengan kata-kata Anda sendiri mengapa hal ini tidak cocok untuk tujuan pengujian perangkat lunak.

(2) Apa tujuan-tujuan lain yang mungkin menggantikan tujuan yang disebutkan di atas, dan apa keuntungan yang dapat diharapkan dalam efektivitas tim pengujian dari perubahan ini?

9.2 Jelaskan dengan kata-kata Anda sendiri mengapa pengujian Big Bang lebih rendah dibandingkan dengan pengujian metode incremental lainnya yang dilakukan untuk paket perangkat lunak yang tidak kecil.

9.3 Modul G12 digabungkan dengan tujuh modul tingkat rendah dan hanya satu modul tingkat tinggi.

(1) Diskusikan bagaimana jumlah kopling mempengaruhi upaya yang diperlukan dalam strategi pengujian incremental.

(2) Pertimbangkan hal yang disebut di atas. Apa dampak dari situasi kopling tertentu dari modul G12 di sumber daya yang dibutuhkan untuk melakukan unit test sesuai dengan strategi top-down dan strategi bottom-up?

9.4 Bagian 9.4 menyebutkan istilah dari cakupan jalur dan cakupan baris.

(1) Jelaskan dengan kata-kata anda sendiri apa syarat utama dan daftar perbedaan utama antara matrik cakupan.

(2) Jelaskan mengapa pelaksanaan dari cakupan jalur adalah suatu hal yang tidak praktis di sebagian besar pengujian aplikasi.

Bengal Wisata adalah agen perjalanan pusat kota yang mengkhususkan diri dalam tur dan liburan di Kanada. Badan ini secara teratur mempekerjakan 25 karyawan tetap. Selama musim semi dan musim panas, badan ini mempekerjakan staf tambahan 20-25 staff sementara, kebanyakan warga senior dan mahasiswa. Badan ini mempertimbangkan untuk membeli hak untuk menggunakan sistem perangkat lunak "Tourplanex", yang mendukung perencanaan mengenai penerbangan dan liburan dan informasi harga. Jika dibeli, perangkat lunak akan menjadi alat kerja utama bagi staff agency.

(1) Diskusikan pentingnya kegunaan pelatihan penggunaan dan kegunaan pengujian operasional yang akan dilakukan oleh agency sebelum pembelian "Tourplanex".

(2) Sarankan kepada manajemen Tours Bengal bahwa mereka harus menerapkan pelatihan penggunaan dan tes penggunaan operasional yang akan dilakukan pada program.

Topik untuk diskusi

9.1 Bhealthy Ltd merupakan sebuah perusahaan asuransi medis yang mengganti biaya obat-obatan dan berbagai biaya medis lain untuk pelanggannya. Menurut prosedur saat ini, pelanggan diminta untuk menunjukkan tanda terima pembelian obat bersama-sama dengan yang resep dokter yang relevan dan dokumen medis lainnya. Pembayaran kembali dihitung sesuai dengan ketentuan perjanjian asuransi:

■Dua daftar obat yang berlaku penggantian: kelas A dan kelas B.

- Kelas A: 90% dari biaya masing-masing obat yang dibeli akan diganti oleh Bhealthy setelah partisipasi pelanggan minimal $ 5. Misalnya, $ 10 obat diganti oleh $ 4,50 dan obat $ 85 adalah diganti dengan $ 72.
- Kelas B: 50% dari biaya masing-masing obat yang dibeli akan diganti oleh Bhealthy (tidak akses).

■ Pemerikasaan akan disusun dan dikirim kepada pelanggan. Perjanjian asuransi negara ini diberikan jangka waktu 45 hari bagi perusahaan untuk menyelesaikan pengembalian uang.

■ Untuk beberapa obat kelas A pelanggan harus memilih untuk membeli obat-obatan sebagai pelanggan swasta yangdiharapkan tidak ada penggantian (harga obat adalah di bawah $ 5).

Penjelasan prosedur terbukti sangat mahal untuk Bhealthy pada saat dan waktu yang sama banyak muncul ketidakpuasan pelanggan . Pertumbuhan jumlah pelanggan serta masalah dengan prosedur membuat Bhealthy termotivasi untuk membuat perjanjian baru dengan apotek berlisensi. Perjanjian tersebut member kewenangan apotek lisensi untuk mengurangi jumlah penggantian dari obat yang berfaktur;Bhealthy kemudian akan mengganti kepada pihak apotek secara bulanan untuk jumlah obat yang dikurangi.

Bhealthy memutuskan untuk menyiapkan sebuah paket software farmasi khusus yang mengkombinasikan operasi penjualan apotek regular dengan operasi yang dibutuhkan perjanjian antara apotek lisensi dan pelanggannya.

Mempertimbangkan modul faktur yang mempersiapkan faktur untuk Bhealthy resep serta untuk penjualan resep reguler dan item lainnya di apotek berlisensi.

(1) Mempersiapkan diagram alir untuk modul.
(2) Siapkan diagram alir program untuk modul.
(3) Hitung kompleksitas cyclomatic untuk modul.
(4) Siapkan set jalur maksimal independen menurut (3). Dokumentasikan jalur dasar dan tunjukkan penambahan tepi setiap jalur independen.

"Sistem Polisi Star 1000" adalah sistem perangkat lunak bergengsi terbaru untuk pencatatan seluruh komunikasi verbal (saluran telepon, telepon seluler dan nirkabel) nasional yang dilaksanakan oleh kepolisian. Salah satu fitur dari sistem ini adalah kemampuannya untuk memasok lengkap setiap rekaman suara dalam 12 bulan terakhir dalam waktu 15 menit di 98% dari aplikasi. Sistem ini direncanakan akan beroperasi dalam waktu 10 bulan.

(1) Diskusikan pentingnya melakukan tes viral komprehensif untuk sistem.
(2) Sarankan anjuran pedoman ini untuk tes viral.
(3) Data dasar apa tentang kegiatan polisi akan Anda rekomendasikan untuk mengumpulkan di perencanaan load test sesuai dengan pedoman saran Anda?

"Super Saving Light" adalah sistem perangkat lunak baru untuk pengendalian penerangan jalan dan peningkatan ekonomi, dikembangkan untuk departemen pemeliharaan kota.
Di antara fungsinya adalah:

■ Memulai dan menyimpulkan dari penerangan jalan sesuai dengan jadwal harian,dijadwalkan setiap tahun.
■ Penerangan sebagian (hanya satu dari setiap dua lampu akan diaktifkan) selama pertama kali dan 15 menit terakhir setiap periode pencahayaan diaktifkan oleh (1).
■ Pengukuran kondisi cahaya alam dengan sensor khusus untuk memastikan apakah pencahayaan alami tidak mencukupi (misalnya, pada hari berawan), yang sebelumnya mengarah kedimulainya penerangan jalan dan kesimpulan mengenai penerangan kemudian. Dalam
kasus-kasus ini, hanya satu dari tiga lampu jalan akan diaktifkan.
■ Pengurangan waktu pencahayaan menurut kepadatan lalu lintas, dipantau oleh sensor lalu lintas dipasang di setiap ruas jalan, yang akan mengurangi penerangan sebagai berikut: jika kepadatan lalu

lintas di bawah satu kendaraan per menit, hanya setengah lampu di bagian jalan yang akan diaktifkan, jika kepadatan lalu lintas kendaraan di bawah 0,3 per menit, hanya sepertiga dari lampu akan diaktifkan.

Tuan Jones, ketua dari tim pengujian, mengklaim bahwa black box tes tidak cukup dan bahwa white box tes diperlukan untuk pengujian "Super Saving Light".

Klaim Tuan Jones didudkung dengan tiga contoh kesalahan perangkat lunak berdasarkan aturan pencahayaan yang dijelaskan di atas. Dalam contoh yang Anda pilih, black box tes akan menghasilkan "OK", sedangkan white box tes dari contoh yang sama akan mendeteksi setidaknya satu kesalahan. Untuk setiap contoh, jelaskan mengapa kesalahan tidak terdeteksi oleh pengujian black box tetapi akan dideteksi dengan pengujian white box.

Berdasarkan kasus "Super Saving Light" yang dijelaskan di atas:

(1) Variabel masukan apa yang diperlukan untuk uji kasus dan apa variable output yang diperlukan?
(2) Sarankan 3 sampai 5 uji kasus sederhana yang memiliki potensi rendah untuk mengidentifikasi kesalahan.
(3) Sarankan 3 sampai 5 uji kasus yang Anda percaya mengandung potensi kesalahan yang serius.
(4) Sarankan 3 sampai 5 uji kasus untuk menghadapi situasi nilai terbatas.

Mengacu pada kasus "Bhealthy" yang dibahas di Topik 9.1, berikut ini adalah daftar harga untuk sampel 12 obat, termasuk kelas penggantian kasus ':
(1) Berdasarkan daftar harga di atas, menyiapkan serangkaian uji kasus yang diperlukan untuk melaksanakan set jalur independen maksimal untuk solusi anda Topic 9.1 untuk pertanyaan (4).
(2) Asumsikan bahwa perubahan partisipasi pelanggan Bhealthy adalah minimum ,untuk kelas obat A dari $ 5 sampai $ 6. Apakah uji kasus dalam (1) harus diubah? Jika ya, buatlah perubahan yang diperlukan dan sajikan update dari file tes kasus tersebut.

Bab10. Penerapan Testing Software

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menggambarkan proses dari perencanaan dan perancangan testing
2. Mendiskusikan sumber dari kasus testing beserta keuntungan dan kerugiannya
3. Mendata tipe utama dari testing software otomatis
4. Mendiskusikan keuntungan dan kerugian dari testing otomatis yang terkomputerisasi sebagai perbandingan dengan testing manual
5. Menjelaskan penerapan testing versi alpha dan beta dan mendiskusikan keuntungan dan kerugiannya.

10.1. Proses Testing

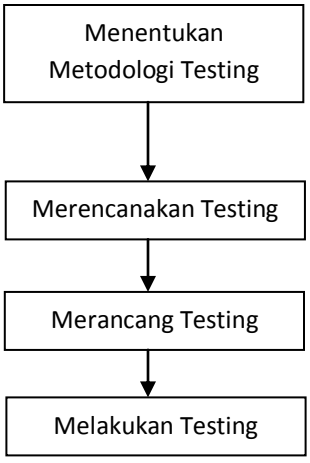
Perencanaan, perancangan dan performa dari testing ditangani seluruhnya dalam proses pengembangan software. Kegiatan ini dibagi menjadi beberapa fase, diawali pada tahap perancangan dan diakhiri ketika software diinstall pada sisi pelanggan. Proses testing ini digambarkan dalam gambar 10.1.

10.1.1. Menentukan Fase Metodologi Testing

Masalah utama dalam harus dihadapi dalam metodologi testing :

- Kesesuaian dengan standar kualitas software yang dibutuhkan
- Strategi testing software

Keputusan tentang dua masalah diatas adalah dasar dan harus dibuat sebelum perencanaan dimulai :



Gambar 10.1. Proses Testing

Menentukan standar kualitas perangkat lunak yang sesuai

Tingkat standar mutu yang dipilih untuk proyek terutama tergantung pada karakteristik dari aplikasi perangkat lunak.

Contoh:

1. Sebuah paket perangkat lunak untuk monitor tempat tidur pasien rumah sakit memerlukan standar perangkat lunak berkualitas tertinggi dengan mempertimbangkan kemungkinan terburuk akibat kegagalan perangkat lunak.
2. Sebuah paket yang dikembangkan untuk menangani informasi umpan balik untuk karyawan internal organisasi program pelatihan dapat dilakukan dengan standar kualitas perangkat lunak tingkat menengah, dengan asumsi bahwa biaya kegagalan relatif rendah (atau jauh lebih rendah dibandingkan dengan Contoh 1).
3. Sebuah paket perangkat lunak telah dikembangkan untuk dijual ke berbagai organisasi luas. Prospek penjualan membenarkan standar mutu yang lebih tinggi dari paket perangkat lunak akan custom-made tidak memiliki kesamaan karakteristik dengan pengembang untuk melayani pelanggan tunggal.

Contoh-contoh ini menggambarkan kriteria utama yang akan diterapkan ketika memilih standar kualitas perangkat lunak: evaluation of nature dan magnitude of expected damages diharapkan kerusakan pada kasus kegagalan sistem. Kerusakan ini dapat mempengaruhi pelanggan dan pengguna di satu sisi, dan pengembang di sisi lain. Secara umum, semakin tinggi tingkat yang diharapkan dari kerusakan akibat kegagalan, semakin tinggi standar kualitas perangkat lunak yang sesuai. Jenis kerusakan tertentu kepada pelanggan dan pengguna serta pengembang tercantum pada Tabel 10.1.

Tabel 10.1. Klasifikasi Kerugian dari Kerusakan Software

a) Kerusakan terhadap pelanggan atau pengguna	
Tipe Kerusakan	Contoh
Membahayakan keselamatan kehidupan manusia	Sistem monitoring pasien rumah sakit
	Sistem persenjataan luar angkasa
Mempengaruhi penggunaan fungsi organisasi yang penting dan tidak ada sistem pengganti	Penjualan sistem e-business
	Sistem inventori multi-ware house
Berakibat terhadap fungsi dari sebuah perusahaan sen	Peralatan elektronik yang terkomputerisasi
Mempengaruhi pemenuhan pengelompokan fungsi tetapi sudah tersedia sebuah pengganti	system penjualan Front-Desk dapat diganti oleh mekanisme manual
Mempengaruhi fungsi dari perangkat lunak untuk aplikasi bisnis	• Tanggapan yang lambat untuk transaksi sebuah system penjualan point-of- sale (POS) • Karena sebuah kesalahan, informasi yang diberikan secara teratur pada satu layar

	didistribusikan antara tiga tampilan yang berbeda
Mempengaruhi berfungsinya piranti perangkat lunak untuk pelanggan khusus	• Game komputer • Pendidikan Pribadi • Word prosesor
Mempengaruhi fungsi dari aplikasi firmware tapi tanpa mempengaruhi	• Aplikasi pengatur dalam rumah tangga tanpa fungsionalitas yang merugikan • Kegagalan sistem sekunder (misalnya, yang menampilkan suhu luar pada system automobile)
Ketidaknyamanan pengguna tetapi tidak mencegah pemenuhan	• Penyimpangan tetapi tidak menyimpang dari tampilan asli

Tabel 10.1. Klasifikasi Kerugian terhadap Pengembang Software

b) Kerugian terhadap pengembang software	
Tipe Kerusakan	Contoh
Kerugian keuangan	Kerusakan yang mengakibatkan luka fisik
	Kerusakan yang dibayar oleh organisasi karena mall fungsi
	Pengembalian uang pembelian dari pelanggan
	Pembiayaan yang tinggi terhadap perbaikan sistem
Kerugian non kuantitatif	Dampak terhadap penjualan mendatang
	Mengurangi penjualan saat ini

Ada beberapa hal yang harus diikutkan dalam menentukan strategi testing software :

- Strategi testing : apakah seharusnya strategi testing big bang atau incremental diadopsi ?
- Bagian mana dari perencanaan testing yang seharusnya dilaksanakan berdasarkan model testing white box ?
- Bagian mana dari perencanaan testing yang seharusnya dilakukan berdasarkan model testing otomatis ?

10.1.2. Perencanaan Testing

Perencanaan testing melibatkan :

- Unit testing
- Integrasi testing
- Sistem testing

Beberapa pertanyaan dalam testing meliputi :

- Apa yang akan ditesting ?
- Source mana yang akan dilakukan sebagai kasus testing ?
- Siapa yang melaksanakan testing ?
- Dimana testing dilaksanakan ?
- Kapan mengakhiri testing ?

Persoalan yang mempengaruhi level resiko software

Persoalan aplikasi /modul :

- Magnitude (besarnya aplikasi)
- Kerumitan dan kesulitan
- Prosentase dari

Untuk apa pengujian diadakan?

Pendekatan langsung untuk pengujian yang sempurna akan merekomendasikan penuh dan perangkat lunak komprehensif dibutuhkan untuk melakukan unit test untuk semua unit individu, integrasi tes untuk semua unit integrasi, dan sistem test untuk menguji paket perangkat lunak secara keseluruhan.

Menerapkan rencana " straightforward " memastikan kualitas perangkat lunak tingkat atas tetapi memerlukan investasi sumber daya yang luas dan jadwal yang diperpanjang. Pertanyaan-pertanyaan tertentu pasti akan timbul sehubungan dengan situasi umum yang berkaitan dengan manfaat dari pendekatan semacam itu. Sebagai contoh:

- Apakah dibenarkan untuk melakukan unit test untuk sebuah modul terdiri dari 98% software yang digunakan kembali?
- Apakah tes unit wajib untuk modul sederhana yang menggambarkan versi ke-12 dari modul dasar yang berulang kali diterapkan oleh tim pengembang selama tiga tahun terakhir?

Hanya dalam kasus yang jarang itu dibenarkan untuk menguji "keseluruhan". Biasanya, kelayakan pengujian "keseluruhan" sangat terbatas. Terlepas dari kinerja daftar tes ditentukan dalam kontrak atau diperlukan oleh prosedur pengembang (misalnya beban tes untuk sistem secara keseluruhan), beberapa pertimbangan urutan preferensi kami untuk tes yang akan diterapkan. Faktor-faktor dalam memutuskan suatu pengujian:

- unit modul yang harus diuji
- integrasi yang harus diuji
- menentukan prioritas pengalokasian sumber daya pengujian kepada individu perangkat lunak

Dalam menentukan apa yang harus dimasukkan dan apa yang dikecualikan dari sistem tes, tes unit dan integrasi yang sudah direncanakan harus dipertimbangkan. Untuk kualitas perangkat lunak aplikasi dan modul yang tidak dicakup oleh unit tes, integrasi dan sistem tes, kami mengandalkan cek kode yang dilakukan oleh programmer dan pemimpin timnya dan inspeksi kode dan walkthrough yang diprakarsai oleh tim pengembang.

Rating unit, integrasi dan aplikasi

Metode untuk unit rating (modul), integrasi dan aplikasi untuk menentukan prioritas dalam rencana pengujian didasarkan pada dua faktor:

- Faktor A: tingkat kerusakan. Tingkat kerusakan hasil dalam kasus modul atau aplikasi gagal.
- Faktor B: tingkat risiko perangkat lunak. Tingkat risiko merupakan probabilitas kegagalan.

10.1.3. Rancangan Testing

Template dari Rencana Testing Software (STP=Software Test Plan) :

1	Lingkup dari testing
1.1	Paket software yang akan ditest (nama, versi dan perbaikan)
1.2	Dokumen yang menyediakan dasar dari perencanaan testing (nama dan versi dari tiap dokumen)

2	Lingkungan testing
2.1	Tempat testing
2.2	Hardware dan konfigurasi organisasi yang dibutuhkan
2.3	Partisipasi organisasi
2.4	Orang yang berkedudukan/berpengaruh yang dibutuhkan
2.5	Persiapan dan pelatihan yang diperlukan oleh tim testing

3	Perincian testing
3.1	Identifikasi testing

3.2	Tujuan testing
3.3	Referensi berseberangan terhadap rancangan dokumen yang relevan dan kebutuhan dokumen
3.4	Kelas testing
3.5	Level testing (unit, integrasi, sistem)
3.6	Kebutuhan kasus testing
3.7	Kebutuhan kusus (pengukuran terhadap response time, kebutuhan keamanan)
3.8	Data yang akan dicatat

4	Jadwal testing (untuk masing-masing test/ group test) termasuk perkiraan waktu
4.1	Persiapan
4.2	Testing
4.3	Perbaikan kesalahan
4.4	Testing mundur

Tes/uji desain dilakukan berdasarkan rencana pengujian perangkat lunak yang didokumentasikan oleh STP. Prosedur Pengujian dan uji kasus database test / file didokumentasikan dalam "Prosedur pengujian perangkat lunak" (software test procedure) dokumen dan " file uji kasus" (test case file) dokumen atau dalam satu dokumen yang disebut " deskripsi test perangkat lunak" (software test description/STD). Sebuah template untuk STD disajikan dalam Frame 10.3.

Frame 10.3 Deskripsi Tes Perangkat Lunak (STD) - template
1 Ruang Lingkup tes
1.1 paket perangkat lunak yang diuji (nama, versi dan revisi)
1.2 Dokumen-dokumen yang menyediakan dasar untuk tes yang dirancang (nama dan versi untuk setiap dokumen)
2 **Lingkup Pengujian** (untuk masing-masing tes)
2.1 Uji identifikasi (rincian tes didokumentasikan dalam STP)
2.2 Keterangan rinci dari sistem operasi dan konfigurasi hardware dan pengaturan switch yang diperlukan untuk tes
2.3 Petunjuk untuk memuat perangkat lunak
3 Proses pengujian
3.1 Petunjuk untuk input, rincian setiap langkah dari proses input
3.2 Data yang akan dicatat selama pengujian
4 uji kasus (untuk setiap kasus)
4.1 Rincian identifikasi uji kasus
4.2 Input data dan pengaturan sistem
4.3 Hasil sementara yang diharapkan (jika berlaku)

4.4 Hasil yang diharapkan (numerik, pesan, aktivasi peralatan, dll)
5 Tindakan yang harus diambil jika terjadi kegagalan program / penghentian
6 Prosedur untuk diterapkan sesuai dengan ringkasan hasil tes

10.1.4. Penerapan Testing

Umumnya, tahap pelaksanaan tes terdiri dari serangkaian tes, koreksi dari kesalahan yang terdeteksi dan re-tes (uji regresi). Pengujian mencapai akhir ketika hasil tes ulang memuaskan para pengembang. Proses tahap implementasi diilustrasikan pada Gambar 10.2. Pengujian dilakukan dengan menjalankan uji kasus sesuai dengan prosedur pengujian. Dokumentasi prosedur pengujian dan uji kasus data-base / file terdiri dari "deskripsi test perangkat lunak" (software test description/STD), disajikan di Frame 10.3.

Template Laporan Testing Software :

1	Testing identifikasi, tempat, jadwal dan partisipan
1.1	Testing identifikasi software (nama, versi, perbaikan)
1.2	Dokumen yang menyediakan dasar testing (nama, versi tiap dokumen)
1.3	Tempat testing
1.4	Waktu mulai dan selesai untuk tiap bagian testing
1.5	Anggota tim testing
1.6	Partisipan yang lain
1.7	Waktu yang diperlukan untuk melakukan testing

2	Lingkungan testing
2.1	Konfigurasi hardware dan perusahaan
2.2	Persiapan dan Pelatihan sebelum testing

3	Hasil testing
3.1	Identifikasi testing
3.2	Hasil kasus testing
	Identifikasi kasus testing
	Identifikasi tester
	Hasil nya : OK atau Gagal
	Jika gagal: gambaran yang jelas dari hasilnya/ permasalahan

4	Tabel kesimpulan untuk total keseluruhan kesalahan, sebarannya dan tipe nya
4.1	Kesimpulan dari testing yang dilakukan
4.2	Perbandingan dengan hasil testing sebelumnya

5	Kejadian khusus dan usulan/pendapat tester
5.1	Kejadian khusus dan tanggapan yang tidak terperkirakan
5.2	Penemuan permasalahan selama testing
5.3	Usulan untuk perubahan lingkungan testing termasuk persiapan testing
5.4	Perubahan untuk perubahan atau perbaikan dalam prosedur testing dan file kasus testing

10.2. Rancangan Kasus Testing

10.2.1. Komponen Data Kasus Testing

Uji kasus adalah sebuah dokumentasi yang terdiri dari masukan data dan kondisi operasi yang dibutuhkan untuk menjalankan sebuah pengujian bersama dengan hasil yang diharapkan. Tester diharapkan dapat menjalankan program untuk pengujian yang sesuai dengan dokumentasi uji kasus, dan kemudian membandingkan hasil aktualnya dengan hasil yang diharapkan yang telah dicatat dalam dokumen. Jika hasil yang diperoleh sepenuhnya sesuai dengan hasil yang diharapkan, tidak ada kesalahan atau setidaknya telah diidentifikasi. Ketika beberapa atau semua hasil tidak sesuai dengan hasil yang diharapkan, diidentifikasi kesalahan potensial. Metode pembagian kesetaraan kelas, dibahas dalam Bagian 9.5.1, diterapkan untuk mencapai efisien dan efektifitas uji kasus, untuk digunakan untuk pengujian kotak hitam.

Contoh
Pertimbangkan uji kasus berikut untuk pajak properti dasar kota tahunan di apartemen. Pajak properti dasar kota (sebelum diskon untuk kelompok-kelompok tertentu dari penduduk kota) didasarkan pada parameter berikut:

S, ukuran apartemen (dalam meter persegi)
N, jumlah orang yang tinggal di apartemen
A, B atau C, klasifikasi pinggiran sosial-ekonomi.

Aplikasi dari kasus testing akan menghasilkan salah satu atau lebih dari tipe hasil yang tidak diharapkan sebagai berikut :

- Urutan angka
- Alfabet (nama, alamat, dll)
- Pesan kesalahan, standar keluaran yang memberikan informasi kepada user tentang kekurangan data, kesalahan data, kondisi yang tidak diharapkan.

10.2.2. Sumber Kasus Testing

Ada dua sumber dasar untuk kasus uji:

- Contoh acak kasus kehidupan nyata. Contohnya:
  - Rumah tangga (untuk menguji sistem informasi pajak kota)
  - Tagihan pengiriman (untuk menguji perangkat lunak penagihan baru)
  - Catatan kontrol (untuk menguji perangkat lunak baru yang mengontrol pembuatan produksi tanaman)
  - Sebuah sampel yang mencatat peristiwa yang akan "dijalankan" sebagai uji kasus (untuk menguji aplikasi online dari situs Internet, dan untuk aplikasi real-time).
- Uji kasus buatan (juga disebut " simulasi uji kasus ") yang disiapkan oleh desainer tes. Jenis ini tidak mengacu pada pelanggan, pengiriman, atau produk tetapi kombinasi dari kondisi sistem operasi dan parameter (didefinisikan oleh satu set data input). Kombinasi ini dirancang untuk mencakup semua operasi perangkat lunak yang dikenal atau setidaknya semua situasi yang diperkirakan sering digunakan atau yang termasuk ke dalam kelas probabilitas kesalahan yang tinggi. Untuk metode kelas ekivalensi, lihat Bagian 9.5.1.

Implikasi menggunakan setiap sumber test case diringkas dan dibandingkan pada Tabel 10.3. Dalam kebanyakan kasus, file uji kasus harus menggabungkan contoh kasus dengan kasus buatan sehingga dapat mengatasi kekurangan satu sumber uji kasus dan untuk meningkatkan efisiensi proses pengujian. Dalam kasus penggabungan file uji kasus, rencana tes sering dilakukan dalam dua tahap: dalam tahap pertama, uji kasus buatan yang digunakan. Setelah koreksi kesalahan terdeteksi, sebuah sampel acak uji kasus digunakan pada tahap kedua.

Tabel 10.3. Perbandingan dari sumber data testing

Keterangan	Tipe Sumber Kasus Testing	
	Random	Buatan/Tiruan
Upaya yang dibutuhkan untuk menyiapkan file	Upaya ringan, khususnya bila keluaran yang diharapan telah diketahui dan tidak memerlukan perhitungan	Upaya besar, parameter dari masing-masing kasus testing harus ditentukan dan keluaran yang diharapkan melalui perhitungan
Ukuran file testing yang dibutuhkan	Relatif tinggi seperti kebanyakan kasus, merujuk pada situasi sederhana yang sering berulang. Dalam rangka untuk mendapatkan jumlah yang cukup dari situasi non-standar, sebuah file uji kasus yang relatif besar perlu dikompile	Relatif kecil dimungkinkan untuk menghindari pengulangan dari setiap kombinasi parameter tertentu
Upaya yang diperlukan untuk melakukan tes perangkat lunak	Upaya Tinggi (efisiensi rendah) tes harus dilakukan untuk file uji	Upaya Rendah (efisiensi tinggi) karena file uji kasus yang

	kasus yang besar. Efisiensi yang rendah berasal dari repetitiveness kondisi kasus, terutama untuk situasi sederhana	dikompilasi relatif kecil untuk menghindari pengulangan
Efektivitas probabilitas deteksi kesalahan	■ relatif rendah - kecuali file uji kasus yang sangat besar - karena rendahnya persentase parameter kombinasi ■ Tidak ada cakupan dari kondisi error ■ Beberapa kemampuan untuk mengidentifikasi kesalahan tak terduga untuk situasi tidak terdaftar	■ Relatif tinggi karena cakupan yang baik ■ Cakupan yang baik dari kondisi error dengan desain file uji kasus ■ kemungkinan kecil untuk mengidentifikasi kesalahan tak terduga karena semua uji kasus yang dirancang sesuai dengan parameter standar

Contoh Bertingkat

Perbaikan substansial dalam efisiensi random sampling uji kasus dicapai dengan menggunakan prosedur sampling yang bertingkat daripada random sampling standar dari seluruh penduduk. Sampling bertingkat memungkinkan kita untuk memecah sampel acak menjadi sub-populasi uji kasus, sehingga mengurangi proporsi mayoritas penduduk "biasa" yang diuji selama meningkatkan proporsi sampling dari populasi kecil dan populasi dengan potensi kesalahan yang tinggi. Metode aplikasi ini meminimalkan jumlah pengulangan pada waktu yang sama yang meningkatkan cakupan dari kondisi yang jarang dan langka.

Sebagai contoh, populasi Garden City sekitar 100 000 rumah tangga terbagi di kota itu sendiri (70%), pinggiran Orange (20%), pinggiran Lemon (7%) dan pinggiran Apple (3%). Pinggiran kota dan kota berbeda secara substansial dalam karakteristik dari rumah mereka dan status sosial-ekonomi. 5% dari rumah tangga, sebagian besar dari mereka penduduk kota, menikmati pengurangan pajak yang melibatkan 40 jenis diskon (cacat, keluarga besar, keluarga single-parent yang berpenghasilan rendah dengan lebih dari enam anak, dll). Awalnya, sampel 0,5% standar telah direncanakan. Kemudian digantikan oleh sampel acak bertingkat berikut:

	Households (no.)	Standard sampling (no.)	Stratified sampling (no.)
Regular households	65 000	325	100
Households enjoying discounts	5 000	25	250
Suburb A	20 000	100	50
Suburb B	7 000	35	50
Suburb C	3 000	15	50
Total	100 000	500	500

Uji kasus untuk perangkat lunak yang digunakan kembali

Hal ini sangat umum untuk perangkat lunak yang digunakan kembali untuk memasukkan banyak aplikasi yang tidak diperlukan sistem perangkat lunak yang ada sekarang di samping aplikasi yang diperlukan. Dalam situasi seperti ini, perencana harus mempertimbangkan yang mana modul perangkat lunak yang digunakan kembali yang harus dites. Modul lain dari perangkat lunak yang digunakan kembali tidak akan diuji.

10.3. Testing Otomatis

Pengujian otomatis merupakan langkah tambahan dalam integrasi alat komputerisasi dalam proses pengembangan perangkat lunak. Alat-alat ini telah bergabung dengan dibantu komputer rekayasa perangkat lunak (CASE) alat dalam melakukan bagian tumbuh analisis perangkat lunak dan tugas-tugas desain.

Beberapa faktor telah memotivasi pengembangan alat pengujian otomatis: penghematan biaya diantisipasi, durasi uji dipersingkat, ketelitian tinggi dari tes yang dilakukan, perbaikan akurasi tes, peningkatan dari hasil pelaporan serta pengolahan statistik dan laporan keuangan. Kemungkinan efisien melakukan berbagai kelas tes yang sebelumnya tidak layak atau tidak mungkin untuk melakukan secara manual, seperti tes viral, juga telah mendorong drive untuk investasi dalam mengotomatisasi pengembangan pengujian.

Sumber berharga untuk bahan tambahan pada pengujian otomatis dapat ditemukan dalam buku seperti Buwalda et al. (2002), Fewster dan Graham (1999) dan Dustin et al. (1999), serta dalam publikasi lain. Bab ini melingkupi :

- Proses otomasi testing
- Tipe dari otomasi testing
- Keuntungan dan kerugian dari otomasi testing

10.3.1. Proses Otomasi Testing

Biasanya, pengujian perangkat lunak otomatis memerlukan perencanaan pengujian, desain uji, persiapan uji kasus, hasil tes, tes log dan persiapan laporan, kembali pengujian setelah koreksi kesalahan yang terdeteksi (uji regresi), dan log ujian akhir dan penyusunan laporan termasuk laporan perbandingan. Dua yang terakhir kegiatan dapat diulang beberapa kal.

Pada tahap perkembangannya, perencanaan, desain dan persiapan ujian kasus pengujian otomatis memerlukan investasi yang besar tenaga kerja profesional. Ini adalah pengujian kinerja komputer dan pelaporan yang menghasilkan ekonomi utama, kualitas dan keunggulan jadwal dari proses tersebut. Ketersediaan tenaga kerja profesional yang dibutuhkan dan sejauh mereka akan digunakan merupakan faktor utama yang harus dipertimbangkan sebelum memulai otomatisasi pengujian perangkat lunak.

10.3.2. Tipe Otomasi Testing

Beberapa jenis pengujian otomatis antara lain :

Code auditing,Pengujian ini memeriksa kesesuaian kode untuk standar tertentu dan prosedur pengkodean.

Coverage monitoring, Bagian ini menghasilkan laporan tentang pencapaian ketika pengimplementasikan file uji kasus tertentu.

Functional tests, Pengujian ini digunakan untuk menggantikan panduan pada black box tes.

Seorang auditor dapat memverifikasi kode berikut:

■ Apakah kode instruksi memenuhi struktur kode dan prosedur?

- Ukuran modul. Beberapa auditor kode hitung untuk kompleksitas kode diuji metrik, seperti metrik cyclomatic McCabe's kompleksitas
- Tingkat loop bersarang
- Tingkat subrutin bersarang
- Dilarang konstruksi, seperti GOTO.

■ Apakah gaya pengkodean mengikuti prosedur gaya pengkodean?

- Penamaan konvensi untuk variabel, file, dll
- Unreachable baris kode program atau subrutin keseluruhan.

■ Apakah dokumentasi program internal dan membantu bagian dukungan mengikuti prosedur pengkodean gaya?

■ Format dan ukuran komentar:

- Lokasi komentar di file

- Bantuan indeks dan gaya presentasi

Cakupan monitoring

Cakupan monitor menghasilkan laporan tentang cakupan garis dicapai ketika mengimplementasikan file uji kasus tertentu. Output monitor termasuk persentase garis yang tercakup dalam kasus uji serta daftar garis ditemukan. Fitur-fitur ini membuat cakupan pemantauan alat vital untuk uji putih-kotak.

Fungsional tes

Tes fungsional otomatis sering mengganti manual tes kotak hitam benar. Sebelum pelaksanaan tes ini, kasus-kasus pengujian dicatat ke database test case. Pengujian kemudian dilakukan dengan menjalankan uji kasus melalui program pengujian. Tes Hasil dokumentasi termasuk daftar dari kesalahan yang diidentifikasi di samping berbagai ringkasan dan statistik sebagai spesifikasi yang dituntut oleh para penguji.

Setelah koreksi telah selesai, kembali menguji seluruh program atau bagian dari itu ("test regresi") umumnya diperlukan. Uji regresi otomatis dilakukan untuk seluruh program memverifikasi bahwa koreksi kesalahan telah dilakukan memuaskan dan bahwa koreksi tidak sengaja diperkenalkan kesalahan baru di bagian lain dari program. Uji regresi sendiri dilakukan dengan database test case yang ada, dengan itu, tes ini dapat dieksekusi dengan sedikit usaha atau sumber daya profesional. Alat uji tambahan otomatis yang mendukung tes fungsional, output komparator, sangat membantu dalam tahap uji regresi. Perbandingan otomatis output dari tes berturut-turut, bersama dengan hasil alat uji fungsional, memungkinkan penguji untuk mempersiapkan analisis peningkatan hasil regresi test dan untuk membantu para pengembang untuk menemukan penyebab kesalahan terdeteksi dalam tes. Hal ini sangat umum untuk program untuk requir tiga atau empat uji regresi sebelum tingkat kualitas dianggap memuaskan.

Load tes

Sejarah pengembangan perangkat lunak sistem berisi bab sedih banyak sistem yang berhasil dalam tes kebenaran tetapi sangat gagal - dan menyebabkan kerusakan besar - setelah mereka diminta untuk beroperasi pada kondisi beban penuh standar. Kerusakan pada banyak kasus sangat tinggi karena kegagalan terjadi "tak terduga", ketika sistem yang seharusnya mulai menyediakan layanan reguler mereka perangkat lunak. Kegagalan paling spektakuler cenderung untuk mengambil tempat dalam sistem informasi yang sangat besar yang melayani sejumlah besar pengguna pada satu waktu atau dalam sistem firmware real-time yang menangani volume tinggi peristiwa simultan.

Untuk tes viral yang harus dilakukan, lingkungan beban maksimal pertama harus diciptakan. Jika dijalankan secara manual, pengujian harus dilakukan ketika sistem berada di bawah beban maksimal pengguna, suatu kondisi yang tidak praktis dalam banyak kasus dan mungkin pada orang lain. Oleh karena itu satu-satunya cara untuk melakukan tes viral untuk sistem menengah dan besar adalah dengan cara simulasi komputer yang dapat diprogram untuk erat pendekatan kondisi beban nyata.

Pengujian beban sendiri didasarkan pada skenario situasi beban maksimal

- terdiri dari kejadian atau transaksi dan frekuensi mereka - bahwa sistem perangkat lunak diharapkan dapat menghadapi dan menangani. Hal ini memungkinkan pengujian otomatis beban (stress test) untuk digabungkan dengan tes ketersediaan dan efisiensi, yang juga membutuhkan lingkungan beban maksimal untuk eksekusi mereka.

Pada titik ini, datang "pengguna virtual dan kegiatan virtual" ke dalam bermain. Untuk skenario operasi diciptakan untuk tes viral, pengguna virtual dan peristiwa virtual dihasilkan dan dioperasikan dalam lingkungan hardware dan komunikasi ditentukan oleh perencana sistem. Seorang pengguna virtual atau peristiwa mengemulasi perilaku pengguna manusia atau peristiwa nyata. Perilaku adalah "dibangun" dengan menerapkan hasil nyata diambil dari aplikasi pengguna nyata, yang kemudian digunakan sebagai masukan untuk simulasi. Beban yang disyaratkan Simulasi dan frekuensi juga diciptakan oleh komputerisasi. Simulasi kemudian menghasilkan output yang mirip dengan yang ditangkap dari pengguna nyata kehidupan di frekuensi dan dengan campuran pengguna didefinisikan oleh skenario. Output ini dapat digunakan sebagai masukan untuk perangkat lunak diuji. Pengujian dilakukan dengan versi yang disetujui akhir perangkat lunak dan dengan hardware yang direncanakan dan konfigurasi komunikasi.

Pemantauan terkomputerisasi tes viral menghasilkan perangkat lunak sistem pengukuran kinerja dalam hal waktu reaksi, waktu proses, dan parameter lain yang diinginkan. Ini adalah dibandingkan dengan persyaratan kinerja beban maksimal yang ditentukan untuk mengevaluasi seberapa baik sistem perangkat lunak akan tampil bila digunakan sehari-hari. Biasanya, serangkaian tes beban dilakukan, dengan beban secara bertahap meningkat menjadi beban maksimal yang ditetapkan dan seterusnya. Langkah ini memungkinkan sebuah studi yang lebih menyeluruh dari kinerja sistem pada kondisi beban penuh. Meja komputer diproduksi dan grafik, berdasarkan informasi pengukuran kinerja, memungkinkan tester untuk memutuskan perubahan apa yang akan diperkenalkan ke setiap simulasi untuk setiap iterasi tes. Sebagai contoh, tester mungkin ingin:

■ Mengubah perangkat keras, termasuk sistem komunikasi, untuk memungkinkan sistem perangkat lunak untuk memenuhi kebutuhan kinerja pada tiap tingkat beban.
■ Mengubah skenario untuk mengungkapkan beban disumbangkan oleh setiap pengguna atau acara.
■ Uji skenario yang sama sekali berbeda
■ Uji baru kombinasi komponen perangkat keras dan skenario.

Tester akan terus beriterasi sampai ia menemukan perangkat keras yang sesuai
konfigurasi.
Contoh:
The "Tick Tiket" adalah situs internet baru yang direncanakan untuk memenuhi persyaratan sebagai berikut:
■ Situs ini harus mampu menangani hingga maksimal 3000 hits per jam.
■ reaksi rata-rata waktu yang dibutuhkan untuk beban maksimal 3000 hits per jam adalah 10 detik atau kurang.
■ reaksi rata-rata waktu yang dibutuhkan untuk beban reguler 1200 hits per jam adalah 3 detik atau kurang.

Rencana: The tes viral direncanakan untuk seri berikut frekuensi hit (hits per jam): 300, 600, 900, 1200, 1500,, 1800 2100, 2400, 2700, 3000, 3300 dan 3600. Sebuah konfigurasi hardware awal ditetapkan, harus disesuaikan menurut hasil tes viral.

Pelaksanaan: Tiga serangkaian tes viral dijalankan sebelum perangkat keras yang memadai dan konfigurasi perangkat lunak komunikasi ditentukan. Setelah seri pertama dan kedua tes viral, konfigurasi hardware telah diubah untuk meningkatkan kapasitas sistem sehingga mencapai waktu reaksi yang diperlukan. Konfigurasi kedua memenuhi persyaratan waktu reaksi untuk beban rata-rata tetapi tidak untuk beban maksimal. Oleh karena itu, kapasitas telah meningkat. Dalam konfigurasi akhir, sistem perangkat lunak yang memuaskan dapat menangani beban 20% lebih tinggi dari pada beban maksimal yang semula ditentukan. Lihat Tabel 10.5 untuk waktu reaksi rata-rata yang diukur pada setiap putaran tes viral.

Uji manajemen

Pengujian melibatkan banyak peserta diduduki secara benar melaksanakan tes dan memperbaiki kesalahan terdeteksi. Selain itu, pengujian biasanya memantau kinerja

dari setiap item pada daftar panjang berkas perkara uji. Beban kerja ini membuat jadwal tindak lanjut penting bagi manajemen. Manajemen tes komputerisasi mendukung ini dan tujuan-tujuan pengujian lainnya manajemen. Secara umum, alat tes komputerisasi manajemen direncanakan untuk menyediakan penguji dengan laporan, daftar dan jenis-jenis informasi pada tingkat kualitas dan ketersediaan yang lebih tinggi dari yang disediakan oleh sistem manajemen manual tes.

Paket perangkat lunak manajemen tes otomatis menyediakan fitur berlaku untuk manual serta pengujian otomatis dan untuk tes otomatis saja. Masukan para penguji kunci dalam, bersama dengan kemampuan paket perangkat lunak, menentukan ruang lingkup aplikasi. Terutama penting di sini adalah interoperabilitas paket dengan sehubungan dengan alat pengujian otomatis.

Frame 10.6 memberikan ringkasan singkat dari fitur yang ditawarkan oleh paket perangkat lunak manajemen tes otomatis.

Ketersediaan alat uji otomatis

Sebagian besar alat-alat pengujian otomatis yang khusus, dan direncanakan untuk digunakan di daerah tertentu aplikasi pemrograman dan sistem: / klien sistem server, C / C + +, UNIX aplikasi, software house tertentu's ERP (Enterprise Resource Planning) aplikasi, untuk mengutip hanya sedikit. Berbagai alat yang saat ini ditawarkan meliputi sebagian wilayah pemrograman yang berlaku dan aplikasi, dan mereka tersedia dari perusahaan pengembangan perangkat lunak yang mengkhususkan diri pada peralatan pengujian otomatis.

10.3.3. Keuntungan dan Kerugian Otomasi Testing

Keuntungan utama menggunakan testing otomatis :

- Akurasi dan kelengkapan pelaksanaan
- Akurasi dari pencatatan hasil dan ringkasan laporan
- Meliputi banyak informasi
- Sedikit sumber daya manusia yang berpengaruh yang dibutuhkan
- Lama testing yang lebih singkat
- Pelaksanaan dari kelengkapan testing
- Pelaksanaan dari kelas testing berdasarkan lingkup dari manual testing

Kerugian utama menggunakan testing otomatis

- Investasi tinggi diperlukan untuk membeli paket dan pelatihan
- Biaya investasi pengembangan paket tinggi
- Membutuhkan orang yang pengetahuan/kewenangan lebih tinggi dalam melakukan persiapan testing.
- Banyak sisa area testing yang tidak tercakup di dalamnya.

10.4. Testing Program Alpha dan Beta

Keuntungan dari testing program beta :

- Identifikasi terhadap keselahan yang tidak diharapkan
- Wilayah pencarian kesalahan yang lebih luas
- Biaya rendah

Kerugian dari testing program beta :

- Kekurangan dari sisi sistematika testing
- Laporan kesalahan yang berkualitas rendah
- Kesulitan menghasilkan lingkungan testing
- Banyak upaya yang dibutuhkan untuk memeriksa laporan.

10.5. Ringkasan

1) **Gambarkan proses dari perencanaan dan perancangan testing !**
2) **Jelaskan sumber-sumber untuk kasus testing serta sebutkan keuntungan dan kerugiaannya !**
3) **Sebutkan tipe utama dari kegiatan testing software secara otomatis !**
4) **Sebutkan keuntungan dan kerugian dari melakukan testing secara otomatis menggunakan komputer dibanding testing secara manual !**
5) **Jelaskan penerapan dari testing alpha dan beta, serta keuntungan dan kerugiannya !**

Bab11. Memastikan Kualitas dari Komponen Pemeliharaan Perangkat Lunak

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Mendata komponen pemeliharaan software dan menjelaskan perbedaannya
2. Menjelaskan dasar dari pemeliharaan yang berkualitas tinggi
3. Menggambarkan dan menjelaskan komponen pre-pemeliharaan software berkualitas
4. Mendata tool infrastruktur yang mendukung pemeliharaan jaminan kualitas
5. Mendata tool pengelolaan untuk mengendalikan kualitas pemeliharaan software dan menjelaskan tingkat kepentingannya.

Dalam bab ini akan menyajikan isu pokok yang terkait dengan pemeliharaan software yaitu :

- Pondasi untuk melakukan pemeliharaan yang berkualitas tinggi
- Komponen kualitas software pada tahap sebelum pemeliharaan
- Tool SQA untuk melakukan pemeliharaan perbaikan
- Tool SQA untuk melakukan pemeliharaan peningkatan fungsional
- Infrastruktur tool SQA untuk perbaikan software
- Pengelolaan pengawasan tool SQA untuk pemeliharaan software.

11.1. Pendahuluan

Tiga komponen pemeliharaan layanan yang sangat penting untuk menunjang kesuksesan adalah :

- Pemeliharaan perbaikan (Corrective maintenance) : user mendukung layanan dan perbaikan software
- Pemeliharaan adaptasi (Adaptive maintenance) : menyesuaikan paket software terhadap kebutuhan pelanggan yang berbeda-beda, adaptasi terhadap kondisi lingkungan yang berbeda dan semacamnya.
- Pemeliharaan peningkatan fungsionalitas : mengkombinasikan (1) perawatan pencegahan terhadap fungsi baru yang ditambahkan terhadap software untuk meningkatkan performa dengan kegiatan pemeliharaan pencegahan yang meningkatkan unsure ketersediaan dan infrastruktur sistem dengan lebih baik dan lebih efisien untuk kemudahan pemeliharaan di masa mendatang

Kesulitan yang dirasakan user mungkin disebabkan oleh :

- Kesalahan kode

- Kesalahan dokumentasi pada buku panduan, layar bantu atau form lain dari sebuah dokumen yang disiapkan untuk user.
- Tidak lengkap, dokumen yang tidak tepat
- User tidak mempunyai pengetahuan yang cukup terhadap sistem software atau kesalahannya dalam menggunakan dokumen yang diberikan.

Tiga penyebab pertama di atas disebut kegagalan sistem perangkat lunak. Selain itu, integrasi layanan dukungan pengguna dan layanan perangkat lunak koreksi umumnya dicapai dalam kerjasama yang erat, dengan berbagi banyak informasi. Komponen lain jasa perbaikan dan pemeliharaan seperti fungsi perbaikan dan pemeliharaan adaptif cenderung tidak akan diprakarsai oleh pengguna layanan dukungan. Dalam kebanyakan kasus, peningkatan fungsi dan tugas adaptif menampilkan karakteristik proyek kecil atau besar, tergantung pada kebutuhan pelanggan. Hal ini yang biasa terjadi, tugas-tugas ini dapat dilakukan oleh unit pengembangan perangkat lunak juga. Menimbang pernyataan di atas, adalah wajar untuk menyertakan layanan dukungan pengguna diantara kegiatan pemeliharaan korektif.

Umumnya, orang dapat mengatakan bahwa untuk sementara pemeliharaan korektif memastikan pengguna saat ini dapat mengoperasikan sistem ini sesuai spesifikasi, pemeliharaan adaptif memungkinkan ekspansi populasi pengguna, sedangkan fungsi peningkatan pemeliharaan memperpanjang masa layanan paket.

Seperti disebutkan dalam bab sebelumnya, kombinasi dari tiga komponen perawatan perangkat lunak mengkonsumsi lebih dari 60% dari total desain dan sumber daya pemrograman yang diinvestasikan dalam sebuah sistem perangkat lunak sepanjang siklus hidupnya (Pressman, 2000). Perkiraan lain menyatakan bagian sumber daya pemeliharaan berkisar dari lebih dari 50% (Lientz dan Swanson, 1980) menjadi sekitar 65-75% (McKee, 1984) sumber daya pembangunan total proyek.

Distribusi sumber daya pemeliharaan atas berbagai jasa pemeliharaan diperkirakan sebagai berikut

Layanan Pemeliharaan	Lientzand and Swanson (1980)	Oskarsson and Glass (1996)
pemeliharaan korektif	22%	17%
pemeliharaan adaptif	24%	23%
Fungsi perbaikan pemeliharaan	54%	60%

Tujuan dari kegiatan penjaminan kualitas dari sisi pemeliharaan software adalah :

- Kepastian, dengan level percaya bahwa kegiatan pemeliharaan perangkat lunak mampu memenuhi kebutuhan fungsional dari sistem
- Kepastian, bahwa dengan aktifitas pemeliharaan perangkat lunak telah mampu selaras dengan kebutuhan jadwal dan biaya

- Kegiatan pengelolaah dan pendahuluan mampu meningkatkan efisiensi dari kegiatan pemeliharaan perangkat lunak. Peningkatan ini termasuk pencapaian fungsional dan pengelolaan serta pengurangan biaya.

11.2. Pondasi dari Kualitas Tinggi

Tak usah dikatakan bahwa mempertahankan kualitas dari paket perangkat lunak mungkin merupakan pondasi yang paling penting yang mendasari kualitas layanan pemeliharaan. Pondasi lain yang penting adalah kebijakan pemeliharaan.

11.2.1. Pondasi 1 : Kualitas Paket Perangkat Lunak

Kualitas dari paket perangkat lunak yang akan dipertahankan jelas berasal dari keahlian dan upaya tim pengembangan serta aktivitas SQA yang dilakukan selama proses pembangunan. Jika kualitas paket adalah jelek, perawatan akan menjadi jelek atau tidak efektif, hampir menurut definisi.

(1) Kebenaran - meliputi:

- **Kebanaran Keluaran**: Kesempurnaan output tertentu (dengan kata lain, tak ada keluaran yang hilang), keakuratan keluaran (output semua sistem diproses dengan benar), yang up-to-datedness output (informasi diolah up to date yang sesuai dengan yang ditetapkan) dan ketersediaan output (waktu reaksi tidak melebihi nilai maksimum yang ditentukan, terutama dalam aplikasi online dan real-time).
- **Dokumentasi Benar**. Kualitas dokumentasi: kelengkapan, akurasi, dokumentasi gaya dan struktur. Format Dokumentasi termasuk hard copy dan file komputer - manual cetak serta elektronik "bantuan" file - bahwa ruang lingkup meliputi instalasi manual, buku petunjuk dan manual programmer.
- **Kualifikasi Koding**. Kepatuhan dengan instruksi coding, terutama membatasi dan mengurangi kompleksitas kode dan mendefinisikan gaya pengkodean standar.

(2) Keandalan. Frekuensi kegagalan sistem serta waktu pemulihan.

Faktor ketiga produk revisi adalah sebagai berikut :

- **Maintainability**. Persyaratan ini dipenuhi pertama dan terutama dengan mengikuti struktur perangkat lunak dan persyaratan gaya dan dengan menerapkan persyaratan programmer dokumentasi.
- **Fleksibility**. Dicapai dengan perencanaan yang tepat dan desain, fitur yang menyediakan ruang aplikasi yang jauh lebih luas daripada yang diperlukan untuk populasi pengguna saat ini. Dalam prakteknya, ini berarti bahwa ruangan yang tersisa untuk perbaikan fungsional masa depan.
- **Testability**. Testability termasuk ketersediaan sistem diagnostik yang akan diterapkan oleh pengguna serta diagnosa kegagalan untuk diterapkan oleh pusat dukungan atau staf pemeliharaan di lokasi pengguna.

Terakhir, faktor produk dua transisi adalah sebagai berikut :

- Portability. Perangkat lunak potensi aplikasi pada hardware yang berbeda dan lingkungan sistem operasi, termasuk kegiatan yang memungkinkan aplikasi tersebut.
- Interoperability. Kapasitas Paket untuk antarmuka dengan paket lainnya dan peralatan terkomputerisasi. Interoperability tinggi dicapai dengan menyediakan kapasitas untuk memenuhi standar interfacing dikenal dan mencocokkan interfacing diterapkan oleh produsen terkemuka peralatan dan perangkat lunak.

Untuk jumlah - upaya untuk menjamin kualitas layanan pemeliharaan harus dimulai awal dalam tahap pengembangan perangkat lunak, ketika masing-masing faktor kualitas ditinjau di atas adalah ditentukan dalam persyaratan proyek dan lagi kemudian, ketika terintegrasi dalam rancangan proyek.
Tujuh faktor di atas dan dampak khas mereka pada berbagai komponen perawatan perangkat lunak disajikan pada Tabel 11.1.

Table 11.1: Quality factors: impacts on software maintenance components

Quality factor	Quality sub-factors	Software maintenance components		
		Corrective	Adaptive	Functionality improvement
Correctness	Output correctness	High		
	Documentation correctness	High	High	High
	Coding qualification	High	High	High
Reliability		High		
Maintainability		High	High	High
Flexibility			High	
Testability		High		
Portability			High	
Interoperability			High	

11.2.2. Kebijakan Pemeliharaan

Komponen utama kebijakan pemeliharaan yang mempengaruhi keberhasilan perawatan perangkat lunak adalah versi pengembangan kebijakan dan perubahan yang akan diterapkan selama siklus hidup perangkat lunak.

Versi pengembangan kebijakan

Kebijakan ini berkaitan terutama untuk pertanyaan tentang bagaimana banyak versi perangkat lunak harus operasi secara bersamaan. Meskipun jelas bahwa ini bukan masalah bagi software custom-made yang melayani satu organisasi, jumlah versi menjadi masalah besar untuk paket-paket perangkat lunak Cots yang direncanakan untuk melayani berbagai macam pelanggan. Pengembangan kebijakan versi terakhir dapat mengambil "berurut" atau bentuk "pohon". Ketika mengadopsi kebijakan versi berurutan, hanya satu versi yang tersedia untuk seluruh pelanggan. Versi ini mencakup profesi aplikasi yang menunjukkan redundansi yang tinggi, atribut yang memungkinkan perangkat lunak untuk melayani kebutuhan semua pelanggan. Perangkat lunak ini harus direvisi secara berkala tapi sekali versi baru selesai, itu menggantikan versi yang saat ini digunakan oleh seluruh pengguna.

Ketika mengadopsi kebijakan versi pohon, tim perawatan perangkat lunak mendukung upaya pemasaran dengan mengembangkan versi, khusus ditargetkan untuk kelompok pelanggan atau pelanggan utama setelah diminta. Sebuah versi baru dilantik dengan menambahkan aplikasi khusus atau menghilangkan aplikasi, tergantung pada apa yang relevan dengan kebutuhan pelanggan. Versi bervariasi dalam kompleksitas dan tingkat aplikasi - aplikasi industri berorientasi target dan sebagainya. Jika kebijakan ini diterapkan, paket perangkat lunak dapat berkembang menjadi sebuah paket multi-versi setelah beberapa tahun kerja, berarti ia akan menyerupai pohon, dengan beberapa cabang utama dan cabang sekunder banyak, masing-masing cabang mewakili sebuah versi dengan revisi khusus. Berbeda dengan versi software sekuensial pemeliharaan, dan pengelolaan perangkat lunak versi pohon jauh lebih sulit dan memakan waktu. Mengingat kekurangan-kekurangan ini, perangkat lunak organisasi-organisasi pembangunan mencoba menerapkan kebijakan pohon versi terbatas, yang memungkinkan hanya sejumlah kecil dari versi perangkat lunak untuk dikembangkan.

Pengalaman harian tim pemeliharaan karena itu termasuk mengatasi kesulitan yang diciptakan oleh struktur versi dari paket yang berkaitan dengan perangkat lunak itu sendiri:

- koreksi kesalahan yang disebabkan oleh identifikasi tidak memadai struktur modul dari versi saat ini digunakan oleh pelanggan tertentu.
- koreksi kesalahan yang disebabkan oleh salah penggantian modul yang rusak dengan modul versi lain yang kemudian terbukti tidak memadai untuk integrasi ke versi paket pelanggan.
- Upaya diinvestasikan untuk meyakinkan pelanggan untuk meng-update paket software mereka dengan menambahkan modul-modul yang baru dikembangkan atau mengganti modul versi saat ini dengan versi yang baru. Setelah upaya berhasil membujuk pelanggan untuk memperbarui paket perangkat lunak mereka, masalah dan kegagalan terjadi ketika mencoba untuk mengintegrasikan modul yang baru dikembangkan atau untuk mengganti saat ini dengan versi lanjutan dari modul.

Perubahan kebijakan

Ubah kebijakan mengacu pada metode pemeriksaan setiap permintaan perubahan dan kriteria yang digunakan untuk persetujuan. Jelas bahwa kebijakan permisif, baik dilaksanakan oleh CCB (the Change Control Board) atau badan lain yang berwenang untuk menyetujui perubahan, memberikan kontribusi untuk peningkatan sering dibenarkan dalam perubahan beban tugas. Kebijakan seimbang, yang memerlukan pemeriksaan menyeluruh terhadap permintaan perubahan, adalah lebih disukai karena memungkinkan staf untuk fokus pada perubahan yang paling penting dan menguntungkan, serta

mereka bahwa mereka akan mampu melakukan dalam waktu yang wajar dan sesuai dengan diperlukan standar kualitas. Kebijakan ini akan berujung pada persetujuan dan hanya sebagian kecil dari permintaan perubahan.

11.3. Komponen Kualitas Software Pre-Pemeliharaan

Seperti komponen SQA pra-proyek, kegiatan pra-perawatan SQA akan selesai sebelum memulai layanan perawatan yang diperlukan adalah sangat penting. Ini memerlukan:

- Pemeliharaan kontrak review
- Pemeliharaan rencana konstruksi.

11.3.1. Pemeliharaan Kontrak Review

Ketika mempertimbangkan kontrak pemeliharaan, perspektif yang luas harus dianut. Apalagi, keputusan yang diperlukan tentang kategori jasa yang akan dikontrak. Keputusan-keputusan ini tergantung pada jenis pelanggan yang dilayani: pelanggan untuk siapa paket custom-made telah dikembangkan, pelanggan yang membeli paket COTS perangkat lunak, dan pelanggan internal. Jadi, sebelum mulai menyediakan jasa pemeliharaan software untuk salah satu pelanggan, kontrak pemeliharaan yang memadai harus diselesaikan yang menetapkan menuruni rentang total kewajiban pemeliharaan sesuai dengan kondisi yang relevan.

Tujuan kontrak pemeliharaan software :

- Persyaratan Klarifikasi Pelanggan
- Pendekatan Alternatif untuk penyediaan pemeliharaan
- Estimasi Sumber Daya perawatan yang dibutuhkan
- Jasa Perbaikan dan pemeliharaan yang akan diberikan oleh sub-contractor dan pelanggan
- Estimasi Biaya Pemeliharaan

11.3.2. Perencanaan Pemeliharaan

Pemeliharaan rencana harus disiapkan untuk semua pelanggan, eksternal dan internal. Rencana ini harus menyediakan kerangka di mana ketentuan pemeliharaan diatur. Oleh karena itu, seperti yang diharapkan, rencana pemeliharaan dan pembangunan (lihat Bab 6) didasarkan pada konsep serupa. Rencana tersebut meliputi:

Rencana pemeliharaan harus disiapkan untuk semua pelanggan , Rencana tersebut meliputi :

- Daftar kontrak layanan pemeliharaan .
- Penjelasan dari organisasi tim pemeliharaan

- Daftar Fasilitas Perawatan
- Daftar identifikasi dari resiko layanan pemeliharaan
- Daftar prosedur pemeliharaan dan kontrol software yang diperlukan
- Anggaran pemeliharaan perangkat lunak

11.4. Tools Jaminan Kualitas Software Pemeliharaan

Berbagai alat besar jaminan kualitas perangkat lunak yang digunakan selama masa theoperational dari siklus hidup perangkat lunak. Sifat khusus dari setiap komponen perawatan perangkat lunak - adaptivemaintenance perbaikan, pemeliharaan dan perbaikan pemeliharaan fungsi - menuntut berbeda set alat SQA digunakan untuk masing-masing. Selanjutnya, operationalperiod software biasanya membuat penggunaan intensif alat SQA infrastruktur dan alat kontrol manajerial lebih mungkin.

Beberapa indikasi tingkat sumber daya yang diinvestasikan dalam SQA selama pemeliharaan telah disiapkan oleh Perry (1995). Dalam survei yang ia dilakukan inNovember 1994, para peserta melaporkan bahwa berdasarkan pengalaman mereka, 31% dari jadwal pemeliharaan mereka didedikasikan untuk jaminan kualitas (review dan tugas pengujian

11.4.1. Tool SQA untuk pemeliharaan kebenaran

Kegiatan pemeliharaan korektif memerlukan terutama (a) layanan dukungan pengguna dan) perangkat lunak koreksi b ((perbaikan bug). Pengguna layanan dukungan menangani kasus-kasus kode perangkat lunak dan dokumentasi kegagalan, atau samar dokumentasi lengkap, mereka juga dapat melibatkan instruksi dari pengguna yang memiliki pengetahuan cukup tentang perangkat lunak atau gagal untuk menggunakan dokumentasi yang tersedia. layanan koreksi Software - perbaikan bug dan koreksi dokumentasi - yang disebut dalam kasus kegagalan perangkat lunak, dan biasanya diberikan selama periode awal operasi (meskipun upaya diinvestasikan dalam pengujian) dan terus diperlukan, meskipun dalam frekuensi yang lebih rendah. Sebagai dua jenis layanan secara inheren berbeda, khas set alat jaminan mutu digunakan terlepas dari bersama berfokus pada kualitas layanan. Meskipun demikian, dalam banyak kasus tim yang sama melakukan kedua jenis pemeliharaan korektif.

Selain pengendalian manajemen SQA dan alat-alat infrastruktur (dibahas kemudian dalam bagian ini), perbaikan bug tugas yang paling membutuhkan penggunaan mini siklus hidup alat SQA, terutama mini-pengujian. Mini-prosedur pengujian yang diperlukan untuk menangani patch perbaikan (skala kecil) tugas, ditandai oleh sejumlah kecil ofcoding perubahan line bersama dengan tekanan kuat untuk menyelesaikan koreksi dengan cepat. Implikasi perbaikan tertunda adalah sedemikian bahwa

mini-singkat - bentuk pengujian sering digunakan. Namun, penggunaan alat-alat mini pengujian ini harus dipertahankan untuk menghindari situasi pengujian tidak ada kompromi. Untuk memastikan " pengujian mini" kualitas, pedoman ini harus dipatuhi:

- Pengujian harus dilakukan oleh tester berkualitas, bukan oleh programmer yang melakukan perbaikan.
- Sebuah prosedur dokumen pengujian (dalam banyak kasus 2-3 halaman panjang) harus disiapkan. Termasuk dalam dokumen adalah deskripsi efek diantisipasi dari perbaikan, lingkup koreksi dan daftar kasus uji yang akan diaktifkan. Prosedur pengujian A-ulang dokumen, mirip dengan dokumen prosedur pengujian, harus juga disiapkan untuk menangani pengujian perbaikan kesalahan terdeteksi dalam tes sebelumnya.
- Sebuah laporan uji mendokumentasikan sepenuhnya kesalahan terdeteksi pada setiap tahap pengujian dan pengujian ulang harus diselesaikan.
- Kepala tim pengujian adalah meninjau dokumentasi pengujian untuk cakupan koreksi, kecukupan kasus uji dan testingresults. Tanggung jawab untuk persetujuan dari perangkat lunak diperbaiki untuk operasional (kadang-kadang disebut "produksi") menggunakan terletak tim kepala.
- Untuk perbaikan dianggap "sederhana dan sepele", terutama bagi mereka tampil di situs pelanggan, mini-pengujian dapat dihindari.

Subkontrak (outsourcing) jasa pemeliharaan, terutama dukungan layanan pengguna, telah menjadi sangat umum setiap kali terlalu merepotkan atau tidak ekonomis untuk kontraktor pemeliharaan untuk langsung memberikan layanan ini. Alat utama untuk menjamin kualitas's pemeliharaan subkontraktor dan membuka jalan bagi hubungan halus adalah kontrak kontraktor-subkontraktor. Alat SQA terintegrasi ke dalam kontrak fokus pada :

- Prosedur untuk menangani berbagai tertentu panggilan pemeliharaan.
- Penuh dokumentasi prosedur pelayanan.
- Ketersediaan mendokumentasikan catatan sertifikasi profesional's pemeliharaan subkontraktor anggota tim, untuk meninjau kontraktor.
- Kuasa untuk kontraktor untuk melaksanakan penelaahan secara periodik terhadap layanan pemeliharaan utama serta survei kepuasan pelanggan.
- Kualitas yang terkait dengan kondisi yang mengharuskan pengenaan denda dan pemutusan kontrak subkontrak dalam kasus yang ekstrim.

11.4.2. Tool SQA untuk pemeliharaan peningkatan fungsionalitas

Karena kesamaan tugas pemeliharaan peningkatan fungsionalitas untuk proyek tugas pengembangan perangkat lunak, hidup alat siklus proyek (review dan pengujian) secara rutin digunakan untuk perbaikan fungsi pemeliharaan. Alat-alat yang sama juga secara rutin diterapkan untuk skala besar tugas-tugas pemeliharaan adaptif, dimana, sekali lagi, karakteristik tugas mirip tugas perbaikan fungsi. Untuk diskusi umum rinci tentang review dan pengujian, lihat Bab 8, 9 dan 10. Tambahan SQA alat diimplementasikan untuk perbaikan pemeliharaan fungsi adalah sistem kontrol manajemen sarana dan prasarana, dibahas kemudian dalam bagian ini.

11.4.3. Komponen infrastruktur SQA untuk pemeliharaan software

Software infrastruktur jaminan kualitas alat, dibahas dalam Bagian IV dari buku ini (Bab 14-19), adalah komponen penting dalam perawatan perangkat lunak. Sebagian besar dari array dari SQA alat infrastruktur yang bersifat umum dan diimplementasikan di seluruh siklus hidup dari sistem perangkat lunak. Selain itu, kesamaan peningkatan fungsi perangkat lunak dan pengembangan proses perangkat lunak memungkinkan kedua proses untuk berbagi SQA sama alat-alat infrastruktur dengan sedikit perubahan. prasarana alat khusus yang diperlukan untuk kegiatan pemeliharaan korektif, karena karakteristik khusus dari kegiatan ini. kegiatan pemeliharaan adaptif dilayani oleh SQA alat infrastruktur, sesuai dengan karakteristik mereka. Sering digunakan alat yang paling yang fungsional SQA perbaikan alat, diikuti oleh SQA pemeliharaan alat korektif.

Sebenarnya, kontribusi SQA alat infrastruktur untuk pemeliharaan tidak dimulai dengan permulaan proses pemeliharaan. Jelas, aplikasi yang memadai alat infrastruktur SQA oleh tim pengembangan perangkat lunak yang memberi kontribusi besar terhadap efisiensi dan efektivitas kegiatan tim pemeliharaan. Dengan kata lain, alat ini memberikan kontribusi terhadap kualitas jaminan pemeliharaan dalam dua cara: pertama, dengan mendukung tim pengembangan perangkat lunak ketika memproduksi perangkat lunak berkualitas tinggi, dan kedua, dengan mendukung tim perawatan bertanggung jawab atas pemeliharaan produk perangkat lunak yang sama.

SQA khusus alat prasarana yang diperlukan untuk proses perawatan perangkat lunak, terutama perawatan korektif, menampilkan karakteristik khusus. Di sini kita fokus pada infrastruktur alat khusus SQA dari kelas berikut:

- Pemeliharaan prosedur dan instruksi kerja
- Mendukung kualitas suatu perangkat / device
- Pelatihan dan sertifikasi team maintenance
- Pencegahan dan tindakan yang hati-hati
- Manajemen konfigurasi
- Dokumentasi dan mengontrol record kualitas

Pemeliharaan prosedur dan instruksi kerja

Kebanyakan prosedur perawatan khusus dan instruksi kerja diterapkan untuk pemeliharaan korektif dan mendukung aktivitas pengguna, misalnya:

- Remote penanganan permintaan untuk layanan dalam kasus-kasus kegagalan perangkat lunak
- Di tempat penanganan permintaan pelanggan untuk layanan dalam kasus-kasus kegagalan perangkat lunak
- Pengguna dukungan layanan
- Jaminan kualitas kontrol koreksi perangkat lunak dan mendukung aktivitas pengguna
- Survei kepuasan pelanggan
- Sertifikasi pemeliharaan korektif dan tim dukungan anggota pengguna.

Pendukung kualitas perangkat

Departemen pemeliharaan diharapkan dapat mengembangkan perangkat khusus untuk mendukung koreksi perangkat lunak dan mendukung aktivitas pengguna: template, daftar dan sejenisnya. perangkat tersebut dapat meliputi:

- Daftar-pembanding untuk lokasi penyebab kegagalan - untuk diterapkan oleh teknisi pemeliharaan.
- Template untuk melaporkan bagaimana kegagalan perangkat lunak tersebut diselesaikan, termasuk temuan dari proses koreksi.
- Daftar-pembanding untuk menyusun prosedur pengujian dokumen mini.

Pelatihan dan sertifikasi tim pemeliharaan

Pelatihan tim pemeliharaan yang berhubungan dengan peningkatan tugas-tugas fungsional tidak berbeda secara substansial dari pelatihan dari tim pengembangan perangkat lunak lain. Namun, pelatihan khusus dan sertifikasi sangat penting untuk tim pemeliharaan korektif.

Pelatihan profesional pemeliharaan korektif dimotivasi oleh kebutuhan untuk menyediakan layanan yang ditentukan dalam kontrak pemeliharaan (atau perjanjian, dalam kasus pelanggan internal) secara berkesinambungan. Dengan demikian, rencana pelatihan harus memberikan solusi untuk kebutuhan staf selama periode beban puncak dan organisasi kebutuhan untuk mengganti, dalam waktu singkat, pensiun, mengundurkan diri atau habis personil. Dalam banyak kasus, pelatihan umum dari cadangan "pemeliharaan personil" tidak cukup, dan pelatihan dalam sistem tertentu harus ditambahkan. Dengan kata lain, program latihan keras yang diperlukan untuk memungkinkan organisasi untuk mengatasi dengan tingkat pelayanan tertentu dikontrak untuk periode beban puncak dan dalam situasi pergantian personil perawatan, karena alasan apapun.

Persyaratan sertifikasi untuk koreksi perangkat lunak dan dukungan personel pengguna berakar pada karakteristik layanan ini. Perhatian khusus harus diberikan untuk sertifikasi profesional koreksi perangkat lunak, yang biasanya melakukan tugas-tugas mereka di bawah tekanan waktu yang berat, bekerja sendiri, dan dalam banyak kasus bekerja di pelanggan situs ini, di mana ketersediaan dukungan profesional dari pemimpin tim atau orang lain terbatas.

Pencegahan dan tindakan korektif

Tahap operasi dari siklus hidup perangkat lunak menghasilkan informasi yang sangat berharga: catatan kegagalan perangkat lunak dan perbaikan mereka serta catatan permintaan dukungan pengguna dapat menyebabkan dan perbaikan tindakan preventif dan ada-dengan berkontribusi terhadap peningkatan perangkat lunak sistem baru dan yang sudah ada. Untuk proses yang akan efektif, perlu ada proses yang memadai untuk screening informasi yang dikumpulkan, mengkaji dan menganalisis temuan, dan menyusun rekomendasi untuk perbaikan pembangunan yang relevan dan proses pemeliharaan. Kegiatan ini SQA diarahkan dan dikendalikan oleh komite internal - dalam CAB (Corrective Action Board), ditemukan dalam organisasi pengembangan perangkat lunak utama. Masalah biasanya disampaikan kepada Dewan untuk diperiksa meliputi:

- Perubahan isi dan frekuensi permintaan pelanggan untuk layanan dukungan pengguna
- Peningkatan rata-rata waktu yang diinvestasikan dalam mematuhi pengguna mendukung permintaan's pelanggan
- Peningkatan rata-rata waktu yang diinvestasikan dalam memperbaiki perangkat lunak kegagalan's pelanggan
- Peningkatan persentase koreksi kegagalan perangkat lunak.

11.4.4. Pengawasan pengelolaan tool SQA untuk pemeliharaan software

Selain digunakan untuk mendapatkan informasi secara berkala, juga digunakan sebagai berikut :

- Software correction
    - Peningkatan penggunaan sumber daya
    - Penurunan tingkat kegagalan pada perbaikan dari jarak jauh
    - Peningkatan tingkat perbaikan on-site di lokasi-lokasi jarak jauh dan jasa luar negeri
- User Support
    - Penigkatan permintaan terhadap layanan suatu software
    - Peningkatan pemanfaatan sumber daya dalam layanan pendukung user
    - Kepuasan pelanggan informasi berdasarkan survei kepuasan pelanggan

11.5. Ringkasan

- Ada tiga komponen perawatan perangkat lunak, masing-masing melakukan :
    - Perbaikan maintenanceperangkat lunak dan layanan pendukung user
    - Adaptive maintenance yang menyesuaikan paket perangkat lunak untuk pelanggan dan perubahan kondisi lingkungan
    - Perbaikan pada fungsi maintenance dengan menggabungkan kegiatan maintenance dengan meningkatkan kinerja dan kehandalan perangkat lunak

- Dua faktor yang dianggap sebagai dasar maintenance yang berkualitas adalah kualitas paket perangkat lunak dan kebijakan maintenance yang diterapkan
- Komponen pra-perawatan kualitas perangkat lunak meliputi (a) review kontrak maintenance dan (b) penyusunan rencana pemeliharaan. Beberapa tujuan kegiatan ini adalah klarifikasi persyaratan pelanggan , analisis terhadap pendekatan alternatif untuk menyediakan layanan dan meninjau estimasi sumber daya dan biaya

1) Sebutkan komponen dari pemeliharaan software dan jelaskan perbedaannya !
2) Jelaskan pondasi dasar dari pemeliharaan yang berkualitas tinggi !
3) Jelaskan dan gambarkan komponen pre-pemeliharaan software yang berkualitas !
4) Sebutkan tool infrastruktur yang mendukung pemeliharaan jaminan kualitas !
5) Sebutkan tool pengelolaan utama untuk mengawasi pemeliharaan software yang berkualitas dan jelaskan tingkat kepentingannya !

Bab12. Memastikan Kualitas melalui Partisipasi Eksternal

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan perbedaan diantara kontraktor dengan partisipan eksternal
2. Mendata tipe dari partisipan eksternal, dan menjelaskan keuntungan mereka menyediakan kontraktor
3. Menggambarkan resiko bagi kontraktor yang tergabung dengan partisipan eksternal
4. Mendata tool SQA yang sesuai untuk digunakan dengan partisipan eksternal dan menambahkan pernyataan pendek dengan memandang resiko yang mereka bisa bantu atau menguranginya

12.1. Tipe dari Partisipan Eksternal

Partner dari proyek pengembangan software – organisasi yang tertarik dengan sistem dari software (pelanggan) dan organisasi yang bertugas melakukan pengembangan (kontraktor) – saat ini seringkali bukan hanya partisipan dalam proyek. Eksternal partisipan terlibat dalam kontribusi pengembangan software dalam proyek tetapi bukan kontraktor, bukan juga kontraktor partner. Kontribusi mereka dalam proyek disusun lewat perjanjian dengan kontraktor (subkontraktor dan pemasok dari COTS perangkat lunak). Semakin besar dan komplek suatu proyek, semakin besar eksternal partisipan akan dibutuhkan, dan semakin besar proporsi pekerjaan yang harus dibagi. Motivasi untuk beralih ke eksternal partisipan bergantung pada beberapa faktor, mulai dari ekonomi ke teknis dan untuk kepentingan personil terkait, dan mencerminkan sebuah tren yang sedang berkembang di alokasi pekerjaan yang terlibat dalam menyelesaikan proyek-proyek kompleks.

Partisipan eksternal dapat dibagi menjadi tiga group utama yaitu :

1) Subkontraktor, biasanya disebut sebagai outsourcing
2) Penyedia dari software COTS dan penggunaan ulang modul software
3) Pelanggan itu sendiri sebagai bagian dalam menyelesaikan sebuah proyek.

12.2. Resiko dan Keuntungan karena Menggunakan Partisipan Eksternal

Resiko yang paling utama terhadap kualitas proyek terkait dengan penggunaan partisipan eksternal dalam sebuah framework proyek meliputi :

1. Penundaan dalam penyelesaian proyek. Saat eksternal partisipan terlambat dalam memasok pekerjaan mereka ke sistem perangkat lunak, maka keseluruhan proyek akan tertunda. Keterlambatan ini merupakan tipikal dari bagian subkontraktor dan bagian pelanggan tapi kurang untuk pemasok software COTS. Dalam banyak kasus pengendalian atas subkontraktor dan pelanggan kewajiban pengembangan software sedikit longgar, situasi seperti ini menyebabkan keterlambatan dan tidak memberikan waktu untuk perubahan dan reorganisasi yang diperlukan untuk menangani penundaan dan membatasi efek negative pada proyek.
2. Rendahnya kualitas dari bagian proyek yang dipasok oleh partisipan eksternal. Masalah kualitas dapat diklasifikasikan sebagai (a) cacat: angka cacat yang lebih tinggi dari yang diperkirakan, sering kali lebih parah dari yang diperkirakan; dan (b) non-standard coding and documentation: pelanggaran style dan struktur dalam instruksi dan prosedur (seharusnya ditetapkan dalam kontrak apapun). Software berkualitas rendah dan non-standard diharapkan untuk menyebabkan kesulitan dalam fase uji coba dan kemudian di tahap pemeliharan. Waktu tambahan yang dibutuhkan untuk menguji dan memperbaiki kualitas yang rendah dari software dapat menyebabkan penundaan proyek bahkan dalam kasus ketika eksternal partisipan menyelesaikan pekerjaan mereka tepat waktu.
3. Kesulitan pemeliharaan pada masa depan. Fakta bahwa beberapa organisasi ambil bagian dalam pembangunan tetapi hanya salah satu dari mereka, kontraktor, bertanggung jawab langsung atas proyek yang membuat dua kemungkinan situasi kesulitan pemeliharaan :
    a. Satu organisasi, biasanya kontraktor, bertanggung jawab atas pemeliharaan seluruh proyek, pengaturan umum ditetapkan dalam tender itu sendiri. Kontraktor mungkin dihadapkan dengan belum selesainya atao non-standard coding dan dokumentasi pasokan oleh eksternal partisipan, menyebabkan rendahnya kualitas pelayanan pemeliharaan yang dilakukan oleh tim pemeliharaan dan biaya yang lebih tinggi pada kontraktor.
    b. Pelayanan perawatan dipasok oleh lebih dari satu organisasi, bisa dari subkontraktor, penyedia COTS software dan kadang-kadang department pengembangan software pelanggan. Masing-masing dari badan-badan tersebut mempunyai tanggung jawab yang terbatas, situasi seperti ini yang memaksa pelanggan untuk mencari tanggung jawab atas kesalahan khusus dari software yang suatu saat ditemukan.

    Kerusakan yang disebabkan oleh kesalahan dari software diharapkan tumbuh di situasi “multi-maintainer”. Baik situasi ini memberikan kontribusi untuk perawatan yang baik dan dapat

diandalkan kecuali langkah-langkah yang memadai diambil sebelumnya, selama tahap perencanaan pengembangan proyek dan pemeliharaan.

4. Kehilangan control atas proyek. Baik disengaja atau tidak, control dari pengembangan software oleh badan-badan eksternal dapat menghasilkan realistis optimis gambar dari suatu proyek. Komunikasi dengan tim eksternal partisipan dapat terganggu selama beberapa minggu, sebuah situasi yang mencegah penilaian dari progress proyek. Akibatnya, peringatan mengenai kesulitan pengembangan, kekurangan staff dan masalah lain yang terlambat sampai ke kontraktor. Kemungkinan untuk solusi tepat waktu dari kesulitan – apakah dengan adaptasi atau perubahan lainnya – dikurangi secara drastic.

Keuntungan bagi kontraktor dalam menggunakan partisipan eksternal :

1) Mengurangi biaya
2) Memperbaiki profesionalisme karyawan
3) Jadwal proyek yang lebih singkat
4) Mendapatkan kemahiran dari para ahli pada bidang tertentu

12.3. Meyakinkan kualitas kontribusi dari partisipan eksternal

Tujuan dibawah ini diturunkan secara langsung dari resiko yang telah kita data sebelumnya :

1) Mencegah keterlambatan penyelesaian tugas dan memastikan tanda awal untuk mengantisipasi keterlambatan.
2) Memastikan level kualitas penerimaan dari bagian yang dikembangkan dan menerima peringatan awal dari pelanggaran terhadap persyaratan kualitas.
3) Memastikan dokumentasinya cukup memadai untuk melayani tim perbaikan
4) Memastikan keberlanjutan, pengetahuan, pengawasan yang dapat diandalkan terhadap performa dari partisipan eksternal

12.4. Tool SQA untuk meyakinkan kualitas dari kontribusi partisipan eksternal

Tool SQA yang diterapkan terhadap partisipan eksternal dalam sebuah proyek pengembangan software :

1) Review terhadap dokumentasi kebutuhan
2) Evaluasi pilihan kriteria terhadap partisipan eksternal
3) Melakukan koordinasi proyek dan membentuk komite pengawasan bersama
4) Berpartisipasi dalam mereview hasil rancangan

5) Berpartisipasi dalam melakukan testing software
6) Memberikan formula terhadap prosedur khusus
7) Sertifikasi dari pimpinan dan anggota tim eksternal
8) Persiapan terhadap laporan kemajuan dari kegiatan pengembangan
9) Review terhadap hasil yang diserahkan (dokumentasi) dan testing penerimaan

12.4.1. Pemeriksaan Dokumen Persyaratan

Tabel 12.1. Daftar persyaratan yang diajukan ke partisipan eksternal :

Tipe Persyaratan	Subyek Persyaratan
Fungsionalitas Software	1) Persyaratan fungsional (terkait kebutuhan pelanggan) 2) Interface antara bagian partisipan eksternal dengan bagian yang lain dalam sebuah proyek 3) Performa, ketersediaan, penggunaan, kehandalan (terkait dengan kebutuhan pelanggan) 4) Layanan pemeliharaan yang diperlukan
Formal dan karyawan	1) Kualifikasi yang dibutuhkan oleh pimpinan dan anggota tim, termasuk sertifikat yang dapat dipakai 2) Melakukan koordinasi dan pembentukan komite bersama (termasuk prosedur untuk menangani keluhan dan persoalan) 3) Daftar dokumen yang diberikan oleh partisipan eksternal 4) Kriteria kelengkapan dari bagian yang dikerjakan partisipan eksternal 5) Rencana penetapan keuangan termasuk bonus dan pinalti
SQA	1) Persyaratan terkait review design dari partisipan eksternal 2) Persyaratan terkait testing software dari partisipan eksternal

12.4.2. Pemilihan Partisipan Eksternal

Tool utama yang dapat digunakan untuk membantuk memilih partisipan eksternal sebagai berikut :

1) Informasi kontraktor tentang supplier dan sub kontraktor berdasarkan pengalaman sebelumnya terkait dengan layanan yang pernah diberikan.
2) Melakukan audit terhadap supplier atau sub kontraktor tentang sistem jaminan kualitas
3) Survey tentang berbagai pendapat yang muncul terkait partisipan eksternal dari sumber luar yang lain.

12.4.3. Koordinasi Proyek dan Komite Pengawasan Bersama

Tujuan utama dari komite ini adalah :

1) Melakukan konfirmasi terhadap timetable (daftar waktu) dan milestone (waktu penting)
2) Melakukan tindakan lanjutan terhadap laporan kemajuan proyek yang dikirimkan ke komite
3) Rapat dengan pimpinan tim dan anggota lainnya dalam suatu bidang dan situasi yang berat

4) Membuat keputusan tentang solusi terhadap timetible dan sumber daya yang muncul selama proyek berlangsung yang telah diidentifikasi dalam tindakan lanjutan.
5) Membuat keputusan terkait solusi dalam mengidentifikasi persoalan dalam review rancangan dan testing software.

12.4.4. Partisipan dalam Pemeriksaan Rancangan

Sejauh mana kontraktor partisipasi diperlukan dalam subkontraktor ' desain review atau ulasan pelanggan kegiatan pembangunan lainnya tergantung pada sifat bagian-bagian proyek yang diberikan oleh para peserta eksternal. Ketika kontraktor berpartisipasi, kita bisa mengharapkan dia untuk bertindak sebagai penuh anggota tinjauan. Dengan kata lain, ia akan membaca dan meninjau dokumen sebelum pertemuan tim dan berpartisipasi dalam diskusi tim serta keputusan yang diambil pada akhir pemeriksaan.

12.4.5. Partisipan dalam Pemeriksaan Software

Partisipasi dalam pengujian perangkat lunak, jika diperlukan, harus mencakup semua tahap dari proses pengujian: tinjauan desain perencanaan dan perancangan tes, review dari hasil pengujian, tindak lanjut pertemuan untuk koreksi dan regresi pengujian. Artinya, karakter partisipasi dalam proses pengujian cukup komprehensif untuk memungkinkan wakil kontraktor untuk campur tangan, jika perlu, untuk mendapatkan jaminan kualitas yang diminta dari disediakan perangkat lunak dan jadwal yang diharapkan untuk menyelesaikan pengujian (dan koreksi) proses.

12.4.6. Prosedur Khusus

Tujuan utama dari prosedur khusus adalah :

1) Persiapan atas persyaratan dokumen bagi partisipan eksternal
2) Pemilihan subkontraktor atau supplier dari COTS software
3) File supplier yang merupakan sumber informasi dan mode dari operasi
4) Perjanjian koordinasi dan komite pengawas bersama untuk bagian-bagian proyek yang diserahkan ke partisipan eksternal dan persiapan petunjuk operasional
5) Persyaratan laporan kemajuan untuk bagian proyek yang diserahkan ke partisipan eksternal.

12.4.7. Sertifikasi dari Pimpinan dan Anggota Tim dari Partisipan Eksternal

Kualifikasi dan sertifikasi dari pimpinan dan anggota tim partisipan eksternal dimaksudkan untuk memastikan tingkat penerimaan pekerjaan secara profesional seperti yang dibutuhkan oleh proyek atau pelanggan. Kebutuhan ini tidak dianggap kecil, kualitas dari anggota tim merupakan inti dari hubungan kerjasama kontrak. Kegiatan SQA yang diperlukan adalah :

1) Kualifikasi dan sertifikasi keahlian anggota tim seharusnya terdaftar sebagai bagian perjanjian kontrak yang diperlukan.
2) Penerapan dari ketentuan dalam kontrak tersebut ditegaskan oleh kontraktor pada permulaan pekerjaan.
3) Perubahan dan penggantian anggota tim yang penting dapat disetujui oleh kontraktor
4) Penerapan dari ketentuan dalam kontrak tersebut secara periodik diperiksa oleh kontraktor.

12.4.8. Laporan Kemajuan

Ketika partisipan eksternal membagi beban kerja proyek, laporan utama kemajuan disiapkan dalam rangka koordinasi dan untuk komite bersama dalam rangka melakukan pengawasan kemajuan, diantaranya sebagai berikut :

- Tindak lanjut dari identifikasi resiko dalam sebuah proyek.
- Tindak lanjut terhadap jadwal proyek.
- Tindak lanjut terhadap penggunaan sumber daya
- Tindak lanjut terhadap penggunaan biaya proyek.

12.4.9. Pemeriksaan terhadap penyerahan hasil (dokumentasi) dan hasil kelulusan testing

Dua dari banyak tool yang sangat bagus untuk memastikan kualitas dari partisipan eksternal, subkontraktor utama dan pelanggan-supplier software adalah melalui pemeriksaan terhadap dokumen pengembangan software oleh kontraktor dan melakukan test penerimaan, perencanaan, perancangan dan diserahkan oleh kontraktor. Tool ini menyediakan cara yang mandiri dan pemeriksaan langsung terhadap dokumen pengembangan pengembangan dan testing dari komponen software dari produk partisipan eksternal.

Oleh karena itu disarankan bahwa kehadiran dari perwakilan subkontraktor dalam tim pemeriksa rancangan dapat mengganti kemandirian pemeriksaan dari produk yang dihasilkan dan testing penerimaan yang dilakukan oleh kontraktor. Dalam banyak kasus, biaya dan jangka waktu merupakan

suatu pertimbangan yang memaksa kontraktor harus dipuaskan dengan pelaksanaan proses testing sistem yang dilakukan oleh subkontraktor. Keputusan tentang pilihan ini seharusnya diambil secara hati-hati.

12.5. Rangkuman

1) Jelaskan perbedaan antara kontraktor dengan partisipan eksternal !
2) Sebutkan tipe dari partisipan eksternal dan jelaskan keuntungan yang mereka sediakan untuk kontraktor
3) Gambarkan resiko bagi gabungan kontraktor terkait dengan peralihan partisipan eksternal !
4) Sebutkan tool SQA yang sesuai untuk digunakan dengan partisipan eksternal dan menambah pernyataan pendek terkait resiko yang mereka bisa bantu baik menghilangkan maupun mengurangi.

Bab 13. Prosedur-Prosedur dan Petunjuk Kerja

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan kontribusi dari prosedur terhadap jaminan kualitas software
2. Menjelaskan perbedaan antara prosedur dan instruksi kerja
3. Mencatat semua kegiatan yang terlibat dalam pemeliharaan pedoman prosedur dari sebuah organisasi.

Prosedur adalah "suatu cara tertentu mencapai sesuatu atau tindakan" (College New Webster's Dictionary). Dengan kata lain, prosedur, seperti yang disebutkan dalam dokumen, adalah kegiatan-kegiatan rinci atau proses yang akan dilakukan menurut metode tertentu untuk tujuan menyelesaikan tugas. The prosedur yang diterapkan oleh organisasi dianggap mengikat bahwa organisasi karyawan, yang berarti bahwa setiap karyawan untuk melakukan nya atau dia tugas-tugas sesuai dengan langkah-langkah yang muncul dalam dokumen prosedur yang relevan, sering membawa nama tugas yang ditunjuk. Prosedur juga cenderung bersifat universal dalam organisasi, yang berarti bahwa mereka akan diterapkan setiap kali tugas dilakukan, terlepas dari orang yang melakukan tugas atau konteks organisasi. Instruksi kerja digunakan terutama dalam kasus di mana sebuah metode seragam melaksanakan tugas di seluruh organisasi adalah baik tidak mungkin atau tidak diinginkan. Akibatnya, instruksi kerja khusus untuk tim atau departemen; mereka suplemen prosedur dengan memberikan rincian eksplisit yang cocok semata-mata untuk kebutuhan satu tim, departemen, atau unit. Prosedur perangkat lunak jaminan mutu dan instruksi kerja khusus bunga kepada kami adalah mereka yang mempengaruhi kualitas sebuah produk perangkat lunak, pemeliharaan perangkat lunak atau manajemen proyek.

Profesional dikembangkan dan dipelihara SQA prosedur yang diperlukan agar sesuai dengan kebijakan mutu organisasi, tetapi juga cenderung sesuai dengan internasional atau nasional SQA standar. Satu hal yang penting untuk diingat ketika mempersiapkan mereka adalah bahwa sesuai prosedural dengan standar SQA mendukung sertifikasi sistem SQA organisasi (lihat Bab VI). The ISO 9000-3 standar (ISO, 1997; ISO / IEC, 2001) adalah salah satu sertifikasi utama standar yang memandu penyusunan prosedur. Smith dan Edge (1991) menyajikan contoh-contoh prosedur.

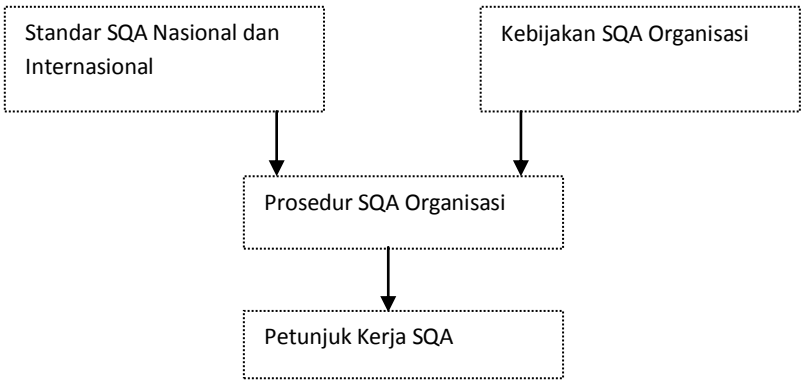
Bab ini akan mendiskusikan tentang :

- ⦿ Kebutuhan akan prosedur SQA
- ⦿ Prosedur dan petunjuk prosedur
- ⦿ Petunjuk kerja dan pedoman petunjuk kerja
- ⦿ Kerangka kerja organisasi dalam rangka persiapan, penerapan dan memperbarui prosedur dan petunjuk kerja.
- ⦿ Menjelaskan kontribusi prosedur untuk jaminan kualitas perangkat lunak
- ⦿ Menjelaskan perbedaan antara prosedur dan instruksi kerja
- ⦿ Dapat membuat daftar kegiatan yang terlibat dalam menjaga prosedur organisasi manual.

13.1. Kebutuhan Terhadap Prosedur dan Petunjuk Kerja

- Mengapa kita seharusnya menggunakan prosedur jaminan kualitas software dan petunjuk kerja ?
- Akankah tidak lebih baik jika setiap profesional menyandarkan pada pengalamannya sendiri dan menunjukkan hasil kerjanya dengan cara terbaik yang dia ketahui ?
- Apa keuntungan dari organisasi dari pemaksaan penyelenggaraan sebuah kerja hanya dengan cara yang diketahui mereka ?

Gambar 13.1 menunjukkan hirarki konsep yang sering digunakan untuk memerintahkan pengembangan prosedur dan petunjuk kerja .



Gambar 13.1. Hirarki Konseptual dari Pengembangan Prosedur dan Petunjuk Kerja

Pertanyaan diatas sering kali ditanyakan oleh beberapa karyawan dalam sebuah organisasi. Jawaban yang melingkupi adalah sebuah tantangan dalam rangkan memenuhi prosedur dan petunjuk kerja. Prosedur SQA dan petunjuk kerja bertujuan untuk :

- Penyelenggaraan tugas, proses maupun kegiatan dengan cara yang paling efektif dan efisien tanpa penyimpangan dari kebutuhan akan kualitas
- Komunikasi yang efektif dan efisien antar karyawan yang terlibat dalam pengembangan dan pemeliharaan sistem software. Seragam dalam penyelenggaraan, dicapai dengan cara mencocokkan dengan prosedur dan petunjuk kerja, mengurangi kesalahpahaman yang dapat memicu kesalahan software.
- Koordinasi yang lebih sederhana antara tugas dan kegiatan yang dilakukan oleh berbagai bagian dari organisasi. Koordinasi yang lebih baik artinya kesalahan yang lebih sedikit.

13.2. Macam Prosedur dan Pedoman Prosedur

Ada lima pertanyaan yang terkait dengan prosedur :

- Kegiatan apa yang harus diselenggarakan ?
- Bagaimana seharusnya masing-masing kegiatan diselenggarakan ?
- Kapan seharusnya kegiatan diselenggarakan ?
- Dimana seharusnya kegiatan diselenggarakan ?
- Siapa yang seharusnya menyelenggarakan kegiatan ?

Isi tetap dari sebuah prosedur adalah :

- Pendahuluan
- Tujuan
- Istilah dan singkatan
- Dokumen yang digunakan
- Metode
- Kualitas catatan dan dokumentasi
- Laporan dan tindakan lanjutan
- Tanggung jawab pelaksanaan
- Daftar apendiks

Pedoman prosedur

Kumpulan prosedur biasanya mengacu pada pedoman prosedur SQA. Isi dari pedoman prosedur berbagai organisasi biasanya terdiri dari :

- Tipe dari pengembangan software dan kegiatan pemeliharaan yang dilakukan sebuah organisasi.
- Range yang dimiliki kegiatan berdasarkan tipe dari masing-masing kegiatan.
- Range dari pelanggan (internal, eksternal) dan supplier
- Konsep yang menentukan pilihan dari metode yang akan diterapkan oleh organisasi untuk mencapai tujuan SQA yang diharapkan.

Sebuah pendekatan yang berguna untuk mendefinisikan struktur dari tabel terhadap isi dari pedoman prosedur SQA digunakan tabel yang berisi standari SQA yang terkait sebagai sebuah kerangka. Tabel 14.1 menampilkan contoh dari penerapan pendekatan ini. Sebagaimana kita lihar, pedoman organisasi membagi prosedur menjadi kategori berdasarkan bab standar ISO yang terkait (dalam tabel, isi dari tabel ditemukan merupakan bagian dari ISO 9000-3 yang digunakan sebagai ilustrasi tujuan ini)

Tabel 14.1. Pedoman prosedur SQA (isi dari tabel)

ISO 9000.3 – isi tabel	Pedoman prosedur SQA – isi tabel
4.1. Tanggung jawab managerial	
4.2. Kualitas Sistem	
4.3. Pemeriksaan Kontrak	
4.4. Pengawasan Rancangan	
4.5. Dokumen dan Pengawasan Data	
4.6. Pembelian/ pembelanjaan	
4.7. Pengawasan terhadap produk dari pelanggan-supplier	
4.8. Identifikasi produk dan rekam jejak	
4.9 Pengawasan Proses	
4.10 Pemeriksaan dan testing	
4.11 Pengawasan pemeriksaan, pengukuran dan testing peralatan	
4.12 Pemeriksaan dan status testing	
4.13 Pengawasan produk yang tidak sesuai	
4.14 Tindakan pencegahan dan perbaikan	
4.15 Penanganan, penyimpanan, pemaketan, perlindungan, penyampaian	
4.16 Pengawasan terhadap catatan kualitas	
4.17 Internal audit kualitas	
4.18 Pelatihan	
4.19 Pelayanan	
4.20 Teknik statistik	

13.3. Petunjuk Kerja dan Pedoman Petunjuk Kerja

Petunjuk kerja departemen :

- Proses audit untuk subkontraktor pengembangan software baru (kandidat supplier)
- Prioritas untuk penanganan tugas perbaikan pemeliharaan
- Evaluasi tiap tahun terhadap subkontraktor pengembangan software
- Petunjuk kerja dan keikutsertaan bagi anggota tim baru
- Template rancangan dokumentasi dan penerapannya
- Petunjuk pemrograman

Petunjuk kerja pengelolaan proyek :

- Koordinasi dan kerjasama dengan pelanggan
- Laporan kemajuan mingguan oleh pimpinan tim
- Template rancangan laporan yang khusus dan penerapannya dalam proyek ini
- Tindak lanjut dari tempat laporan versi beta
- Laporan kemajuan bulanan ke pelanggan
- Koordinasi tentang installasi dan petunjuk kepada tim pelanggan

13.4. Prosedur dan Petunjuk Kerja : Persiapan, Penerapan dan Pembaruan

Motivasi untuk memperbarui prosedur yang ada berdasarkan hal dibawah ini :

- Perubahan teknologi dalam toop pengembangan, hardware, peralatan komunikasi dll
- Perubahan terhadap wilayah kegiatan organisasi
- Proposal user terhadap peningkatan
- Analisis terhadap kegagalan maupun keberhasilan
- Proposal untuk peningkatan yang diprakarsai oleh laporan audit internal
- Belajar dari pengalaman dari organisasi lain
- Pengalaman dari tim SQA.

13.5. Ringkasan

1) Jelaskan kontribusi dari prosedur tehadap jaminan kualitas software !

2) Jelaskan perbedaan antara prosedur dan petunjuk kerja !

3) Sebutkan kegiatan yang terlibat dalam pemeliharaan pedoman prosedur yang dimiliki sebuah organisasi !

Bab14. Peralatan Pendukung Kualitas

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan kontribusi utama dari template terhadap jaminan kualitas software
2. Menjelaskan kontribusi utama dari cek list terhadap jaminan kualitas software
3. Mendata kegiatan yang terlibat dalam pemeliharaan template dan cek list

14.1. Template

Tiga template yang akan disajikan adalah :

- Frame 10.2 : Software Test Planning (STP)
- Frame 10.3 : Software Test Description (STD)
- Frame 10.4 : Software Test Report (STR)
- Frame 18.4 : Software Change Request (SCR)
- Frame 18.6 : Dokumentasi dari Konfigurasi Software yang dikeluarkan

14.1.1. Kontribusi dari template terhadap kualitas software

Penggunaan template sangat menguntungkan untuk mengembangakan dan melakukan pemeriksaan terhadap tim. Untuk pengembangan tim, template digunakan untuk :

- Memfasilitasi proses penyiapan dokumen
- Memastikan bahwa dokumen yang dipersiapkan pengembang lebih dari sekedar lengkap.
- Menyediakan cara integrasi yang mudah bagi anggota tim baru
- Memfasilitasi pemeriksaan dokumen dengan cara mengurangi waktu mempelajari struktur dokumen dan menetapkan kelengkapannya.

Untuk tim pemeliharaan software, template digunakan untuk :

- Memudahkan mencari lokasi dari informasi yang diperlukan dalam rangka melakukan pemeliharaan.

14.1.2. Pengorganisasian framework terhadap template persiapan, penerapan dan pembaruan

Persiapan Template Baru

Sumber informasi utama yang pada umumnya digunakan untuk menyiapkan sebuah template adalah :

- Template informasi yang telah digunakan dalam organisasi
- Contoh template yang ditemukan dalam publikasi yang digunakan secara profesional
- Template yang digunakan organisasi sejenis

Template Aplikasi

Beberapa keputusan dasar yang terlibat dalam penerapan beberapa template baru atau pembaruan template adalah :

- Saluran apa yang seharusnya digunakan untuk pemasangan iklan bagi template ?
- Bagaimana caranya agar template tersedia bagi pengguna organisasi internal ?
- Template mana yang wajib dan bagaimana aplikasinya diterapkan ?

Updating template

Keputusan untuk memperbarui sebuah template dipertimbangkan sebagai sebuah ukuran yang reaktif, isi pokoknya sebagai berikut :

- Proposal dan saran dari pelanggan
- Perubahan wilayah kegiatan organisasi
- Proposal yang diprakarsai oleh pemeriksaan rancangan dan pemeriksaan tim berdasarkan pemeriksaan mereka terhadap dokumen yang disiapkan berdasarkan template-template yang ada.
- Analisa kegagalan yang sebaik analisa keberhasilan
- Pengalaman organisasi lain
- Prakarsa tim SQA.

14.2. Cek List

Nama Perusahaan					
Daftar Periksa Untuk Laporan Spesifikasi Kebutuhan					
Nama Proyek					
Dokumen Periksa			Versi		
No	Subyek	Ya	Tidak	TT*	Komentar
1	Dokumen				
1.1.	Persiapan berdasarkan kebutuhan pengelolaan konfigurasi				
1.2.	Penyesuaian struktur berdasarkan template yang relevan				
1.3.	Dokumen periksa telah lengkap				
1.4.	Referensi dokumen yang sesuai untuk membentuk dokumen, standarisasi dll				

2	Penetapan Kebutuhan				
2.1.	Fungsional yang dibutuhkan terdefinisi dengan tepat dan penuh dengan ungkapan yang jelas				
2.2.	Rancangan input telah sesuai dengan kebutuhan output				
2.3.	Spesifikasi kebutuhan software telah sesuai dengan kebutuhan produk				
2.4.	Kebutuhan interface dengan paket software eksternal dan peralatan yang terkomputerisasi telah didefinisikan sepenuhnya, dan penuh dengan ungkapan yang jelas				
2.5.	Interface GUI telah didefinisikan dengan jelas serta dengan ungkapan yang jelas				
2.6.	Pelaksanaan kebutuhan (waktu tanggap, kapasitas aliran input, kapasitas penyimpanan, didefinisikan dengan jelas serta dengan ungkapan yang jelas pula				
2.7.	Segala kondisi yang salah dan membutuhkan reaksi sistem didefinisikan dengan benar dan diungkapkan dengan jelas				
2.8.	Interfaces data dengan rancangan paket software yang lain atau yang telah ada atau dengan komponen produk lain telah didefinisikan dan diungkapkan dengan jelas				
2.9.	Prosedur untuk melakukan pemenuhan testing terhadap kebutuhan yang telah ditetapkan telah didefinisikan dan diungkapan dengan jelas				
3	Kemungkinan kejadian proyek				
3.1.	Apakah semua kebutuhan yang telah ditetapkan dengan mudah dapat dikerjakan berdasarkan atas sumber daya proyek, biaya dan waktu yang tersedia				
3.2.	Apakah penyelenggaraan kebutuhan yang telah ditetapkan mungkin dikerjakan berdasarkan batasan yang diadakan oleh sistem komponen lain dan oleh interface sistem lain dengan sistem ?				

Komentar :

Tanda Tangan : Tanggal : Nama :

TT* : Tidak Terpakai

14.2.1. Kontribusi Cek List terhadap kualitas software

Seperti template, cek list menyediakan beberapa keuntungan untuk mengembangkan tim, tim pemeliharaan software dan kualitas dokumen. Keuntungan pengembangan tim adalah sebagai berikut :

- Membantu pengembang membawa hasil pemeriksaan dokumen atau koding software sebelumnya ke tahap kelengkapan dokumen atau kode software dan pemeriksaan rancangan formal.
- Membantu pengembang dalam persiapan tugas mereka seperti instalasi software di tempat pelanggan

Keutungan untuk melakukan pemeriksaan terhadap tim sebagai berikut :

- Memastikan kelengkapan dari dokumen pemeriksaan oleh anggota tim pemeriksa terhadap semua item pemeriksaan yang muncul dalam daftar pemeriksaan.
- Memberikan fasilitas untuk peningkatan efisiensi dari tahap pemeriksaan sebagai subyek dan permintaan diskusi yang didefinisikan dan dikenal baik dalam tahap lanjut.

14.2.2. Pengorganisasian framework terhadap cek list tahap persiapan, penerapan dan pembaruan

Persiapan cek list yang baru dalam rangka peningkatan dari cek list yang informal harus didukung oleh beberapa sumber informasi diantaranya :

- Cek list informal yang telah ada dan digunakan oleh organisasi.
- Contoh cek list yang ditemukan di buku dan publikasi profesional lain
- Cek list yang digunakan organisasi serupa

Pembaruan ceklist

Seperti template dan prosedur, prakarsa untuk memperbarui cek list yang ada pada umumnya mengalir dari sumber berikut :

- Proposal dan saran dari pelanggan
- Perubahan wilayah kegiatan organisasi
- Proposal yang diprakarsai oleh pemeriksaan rancangan dan pemeriksaan tim berdasarkan pemeriksaan mereka terhadap dokumen yang disiapkan berdasarkan template-template yang ada.
- Analisa kegagalan yang sebaik analisa keberhasilan
- Pengalaman organisasi lain
- Prakarsa tim SQA.

14.3. Rangkuman

1) Sebutkan kontribusi dari template terhadap jaminan kualitas software !
2) Jelaskan kontribusi utama dari ceklist untuk jaminan kualitas software !
3) Sebutkan aktifitas yang terlibat dalam pemeliharaan template dan cek list !

Bab15. Pelatihan Karyawan dan Sertifikasi

Sesudah mempelajari bab ini, diharapkan kita mampu :

1. Menjelaskan tujuan utama dari pelatihan dan sertifikasi
2. Menjelaskan apa yang dibutuhkan untuk menyiapkan pelatihan dan pembaruan program
3. Mendata komponen utama pada program sertifikasi
4. Menjelaskan tujuan tindak lanjut pelatihan dan program sertifikasi karyawan dan sumber utama untuk melakukan tindak lanjut.

15.1. Pendahuluan

Bisa dikatakan bahwa memiliki staf yang memahami pengetahuan terkini adalah merupakan kunci untuk mencapai kualitas pengembangan dan pemeliharaan. Pada umumnya, hal ini diperoleh dengan melakukan pelatihan rutin secara professional, pelatihan ulang dan update pengetahuan adalah hal yang wajib dilakukan untuk menjaga jarak antara pengetahuan sekarang dan pengetahuan professional yang dibutuhkan.

15.2. Tujuan dari Pelatihan dan Sertifikasi

Tujuan dari pelatihan dan sertifikasi :

- Mengembangkan pengetahuan dan ketrampilan karyawan yang diperlukan untuk mengembangkan software dan tugas pemeliharaan pada level efektifitas yang sesuai. Beberapa pelatihan dilakukan terhadap anggota tim baru.
- Memastikan kesesuaian standar organisasi terhadap produk software (dokumen dan koding) dengan cara mengirimkan mode dan struktur bersama dengan petunjuk kerja.
- Memperbarui pengetahuan dan keahlian dari karyawan yang lama sebagai tanggapan terhadap pengembangan organisasi dan memastikan efisiensi terhadap pekerjaan yang memenuhi kesesuaian mode dan struktur prosedur dan petunjuk kerja.
- Meneruskan pengetahuan tentang prosedur SQA
- Memastikan kandidat sebagai kunci personal untuk pengembangan software dan posisi pemeliharaan yang sesuai dan memenuhi syarat.

15.3. Proses Pelatihan dan Sertifikasi

Proses dari pelatihan dan sertifikasi yang berhasil membutuhkan kegiatan yang reguler sebagai berikut :

- Menentukan kebutuhan pengetahuan profesional untuk masing-masing profesi.
- Menentukan pelatihan profesional dan kebutuhan yang terbaru
- Merencanakan program pelatihan profesional
- Merencanakan program pembaruan profesional
- Menentukan posisi sertifikasi yang dibutuhkan
- Merencanakan proses sertifikasi
- Melaksanakan pelatihan, pembaruan dan program sertifikasi.
- Melakukan tindak lanjut pelatihan dan sertifikasi

REFERENSI

i. Daniel Galin, “Software Quality Assurance”, PEARSON Addison Wesly, 2004

ii. Evan W. Duggan and Han Reichgelt, “Measuring Information System Delivery Quality”, IDEA GROUP PUBLISHING, 2006